DATA SCIENCE

Facualty of Computers and Information
Beni-Suef University

TRAFFIC and ACCIDENTS ANALYSIS

Graduation project

Team Work

1. Muhamed Gamal Mahmoud id:14 -0103
2. Kareem Rabea Mossad id:14 -0093
3. Muhamed Ahmed Ibrahim id:14 -0098
4. Ahmed Shaban Saad id:14 -0009
5. Muhamed Mokhtar Rabea id:14 -0121

Under Supervision

Dr/Kareem Kamal

Data Science for Traffic Flow and Accidents Problems in UK.

Short project Description:

- Using data science technology to get the information from Data, insights and solve related problems to make smart decisions based on real facts.
- Design user interface for organization to can get Information and insights by itself.
- Using machinge learning techniques And Statistics to predict accidents rate over time.

Index

Table of figures

Chapter three

Chapter four

Chapter five

Chapter six

Chapter seven

Chapter eight

Abstract

Data science, also known as data-driven science, is an interdisciplinary field of scientific methods, processes, and systemstoextract knowledge or insights from data in various forms, structured or unstructured, similar to data mining.

Data science is a "concept to unify statistics, data analysis and their related methods" in order to "understand and analysis actual phenomena" with data.

It employs techniques and theories drawn from many fields within the broad areas of mathematics, statistics, information science,and computer science, inparticular from the subdomains of machine learning, classification, cluster analysis, uncertainty quantification, computational science, datamining, databases, and visualization.

The purpose of our project is to get knowledge from dataset of traffic flow of UK to solve problems of road accidents and ways how to reduce percentage of accident or avoid reasons which lead to accidents and we provide an application to help in analytic datasets and get knowledge from it as graphs and predict the rate of accidents over the next years.

1

chapter one

introduction

- motivation
- problem
- Aims of the project
- releated Work

1.1 Motivation

Every day and every moment increase number of vechiles in the world,via this growing, the probability of occurring accidents grows too.

This growing everyday can affect on life of people negatively or positively. So we are going to study this phenomenon of traffic flow and accidents in UK.

Data science is very important and new technique helping companies and orgranizations in statical field and get results to solve problems.

Due to the high incidence of accidents in UK and a large number of injuries and deaths due to these accidents, it is necessary to find solutions to reduce the proportion of these incidents by knowing the reasons that lead to the occurrence of an accident. We use the data science field for analysis data and making predictive modelling.

1.2 Problem

1.2.1 Context

The UK government amassed traffic data from 2000 and 2016, recording over 1.6 million accidents in the process and making this one of the most comprehensive traffic data sets out there. It's a huge picture of a country undergoing change.

Note that all the contained accident data comes from police reports, so this data does not include minor incidents. (Government, 2017)

1.2.2 Content

UkTrafficAADF.csv tracks how much traffic there was on all major roads in the given time period (2000 through 2016). AADT, the core statistic

included in this file, stands for "Average Annual Daily Flow", and is a measure of how activity a road segment based on how many vehicle trips traverse it. The AADT page on Wikipedia is a good reference on the subject.

Accidents data is split across three CSV files: accidents_2005_to_2007.csv, accidents_2009_to_2011.csv, and accidents_2012_to_2014.csv. These three files together constitute 1.6 million traffic accidents. The total time period is 2005 through 2014, but 2008 is missing.

A data dictionary for the raw dataset at large is available from the UK Department of Transport website. For descriptions of individual columns, see the column metadata.

UkTrafficAADF.csv tracks how much traffic there was on all major roads in the given time period (2000 through 2016). AADT, the core statistic included in this file, stands for "Average Annual Daily Flow", and is a measure of how activity a road segment based on how many vehicle trips traverse it. The AADT page on Wikipedia is a good reference on the subject.

Accidents data is split across three CSV files: accidents_2005_to_2007.csv, accidents_2009_to_2011.csv, and accidents_2012_to_2014.csv. These three files together constitute 1.6 million traffic accidents. The total time period is 2005 through 2014, but 2008 is missing.

A data dictionary for the raw dataset at large is available from the UK Department of Transport website. For descriptions of individual columns, see the column metadata. (Government, 2017)

1.2.4 Goals

- How has changing traffic flow impacted accidents?
- Predict accident rates over time? What might improve accident rates?

- Plot interactive maps of changing trends, e.g. How has London has changed for cyclists? Busiest roads in the nation?
- Areas never change and why? Identify infrastructure needs, failings and successes.
- How have Rural and Urban areas differed (see RoadCategory)? How about the differences between England, Scotland, and Wales?
- The UK government also like to look at miles driven. You can do this by multiplying the AADF by the corresponding length of road (link length) and by the number of days in the years. What does this tell you about UK roads?

1.3 Aims of the Project

- The project aim to analysis data of accidents and traffic flow.
- Get the insights and information from data.
- Making predictive modelling using machine learning.
- Develop web shiny.

2

chapter two

Data description

- Traffic Data description
- Accidents Data description

2.1 Traffic Data Description

- AADFYear = The year data was recorded
- CP = Count point. A unique reference for the road link that links the AADFs to the road network
- Estimation_method = Counted or estimated
- Estimation_method_detailed = How it was counted, or how it was estimated
- Region = 11 regions, essentially Wales, Scotland and then 9 English regions. No data from N.Ireland.
- LocalAuthority = Breaking down the regions further
- Road = Roads are either M for Motorway (the biggest roads), or A which are our major roads. B roads aren't included
- RoadCategory = See details below*
- Easting = Cartesian X co-ordinate (lat and lon are in the last two columns)
- Northing = Cartesian Y co-ordinate (lat and lon are in the last two columns) * Easting and Northing need to use epsg=27700. Explained in the Kernel 'Using Basemap for geographic data'
- StartJunction = Where the stretch of road begins
- EndJunction = Where the stretch ends
- LinkLength_km = How far between the start and end
- LinkLength_miles = How far between the start and end
- PedalCycles = Avergae volume per day
- Motorcycles = Avergae volume per day
- CarsTaxis = Avergae volume per day
- BusesCoaches = Avergae volume per day
- LightGoodsVehicles = Avergae volume per day
- V2AxleRigidHGV = HGV is a Lorry/Truck. It stands for Heavy Goods Vehicle. The number of Axles lets you know how big it is
- V3AxleRigidHGV
- V4or5AxleRigidHGV

- V3or4AxleArticHGV
- V5AxleArticHGV
- V6orMoreAxleArticHGV
- AllHGVs
- AllMotorVehicles
- Lat = Latitude
- Lon = Longitude

2.2 Road Category Description

- PM = M or Class A Principal Motorway
- PR = Class A Principal road in Rural area
- PU = Class A Principal road in Urban area
- TM = M or Class A Trunk Motorway
- TR = Class A Trunk road in Rural area
- TU = Class A Trunk road in Urban area

2.3 Information about Miles Driven

- The UK governments also like to look at miles driven. You can do this by multiplying the AADF by the corresponding length of road (link length) and by the number of days in the years.

2.4 Accidents Data Description

- Accident_Index = Unique ID
- Location_Easting_OSGR = Local British coordinates
- Location_Northing_OSGR = Local British coordinates
- Longitude
- Latitude
- Police_Force
- Accident_Severity = The higher the number, the worse it is
- Number_of_Vehicles
- Number_of_Casualties

- Date = dd/mm/yyyy
- Day_of_Week = 1 is Monday, 2 is Tuesday, etc
- Time = GMT/UTC (Grenwich Mean Time and Coordinated Universal Time are the same)
- Local_Authority_(District)
- Local_Authority_(Highway)
- 1st_Road_Class = This field is only used for junctions
- 1st_Road_Number = This field is only used for junctions
- Road_Type = Roundabout, One way, Dual Carriageway, Single carriageway, slip road, unknown
- Speed_limit
- Junction_Detail = Crossroads, roundabouts, private roads, not a junction, etc
- Junction_Control = A person, a type of sign, automated, etc.
- 2nd_Road_Class = This field is only used for junctions
- 2nd_Road_Number = This field is only used for junctions
- Pedestrian_Crossing-Human_Control = Was there a human controller and what type?
- Pedestrian_Crossing-Physical_Facilities = was it a zebra crossing, or bridge, or another type
- Light_Conditions = Day, night, street lights or not
- Weather_Conditions = Wind, rain, snow, fog
- Road_Surface_Conditions = Wet, snow, ice, flood
- Special_Conditions_at_Site = Was anything broken or defective, e.g. an obscured sign?
- Carriageway_Hazards = Was something in the way, e.g. a pedestrian, another accident, something in the road?
- Urban_or_Rural_Area
- Did_Police_Officer_Attend_Scene_of_Accident
- LSOA_of_Accident_Location

2.5 Missing Data

- 2008 is missing from accident file
- South East is missing Estimation method from traffic file

3

chapter three

Tools

- *Whate is R?*
- *R-studio*
- *Tidyvers*
- *Other Packages*

Tools

There are four things you need to run the code: R, RStudio, a collection of R packages called the tidyverse, and a handful of other packages. Packages are the fundamental units of reproducible R code. They include reusable functions, the documentation that describes how to use them, and sample data.

3.1 R

3.1.1 History:

R is an implementation of the S programming language combined with lexical scoping semantics inspired by Scheme. S was created by John Chambers in 1976, while at Bell Labs. There are some important differences, but much of the code written for S runs unaltered.

R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team, of which Chambers is a member. R is named partly after the first names of the first two R authors and partly as a play on the name of S. The project was conceived in 1992, with an initial version released in 1995 and a stable beta version in 2000.

3.1.2 Statistical Features:

R and its libraries implement a wide variety of statistical and graphical techniques, including linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering, and others. R is easily extensible through functions and extensions, and the R community is noted for its active contributions in terms of packages. Many of R's standard functions are written in R itself, which makes it easy for users to follow the algorithmic choices made. For computationally intensive tasks,

C, C++, and FORTRAN code can be linked and called at run time. Advanced users can write C, C++, Java, .NET or Python code to manipulate R objects directly. R is highly extensible through the use of user-submitted packages for specific functions or specific areas of study. Due to its S heritage, R has stronger object-oriented programming facilities than most statistical computing languages. Extending R is also eased by its lexical scoping rules.

Strength of R is static graphics, which can produce publication-quality graphs, including mathematical symbols. Dynamic and interactive graphics are available through additional packages.

R has Rd, its own LaTeX-like documentation format, which is used to supply comprehensive documentation, both on-line in a number of formats and in hard copy. (Wickham, 2017)

3.1.3 Downloading:

To download R, go to CRAN, the comprehensive R archive network. CRAN is composed of a set of mirror servers distributed around the world and is used to distribute R and R packages. Don't try and pick a mirror that's close to you: instead use the cloud mirror, https://cloud.r-project.org, which automatically figures it out for you.

A new major version of R comes out once a year, and there are 2-3 minor releases each year. It's a good idea to update regularly. Upgrading can be a bit of a hassle, especially for major versions, which require you to reinstall all your packages, but putting it off only makes it worse.

3.2 R-studio

RStudio is an integrated development environment, or IDE, for R programming.

Download and install it from http://www.rstudio.com/download. RStudio is updated a couple of times a year. When a new version is available, RStudio will let you know. It's a good idea to upgrade regularly so you can take advantage of the latest and greatest features. For this book, make sure you have RStudio 1.0.0.

When you start RStudio, you'll see two key regions in the interface:

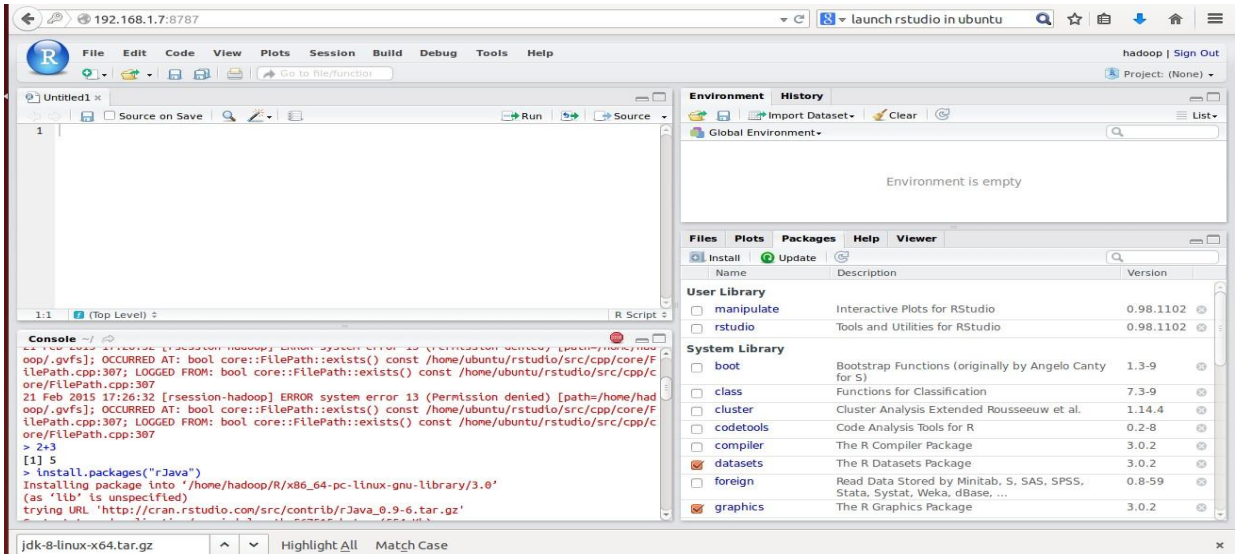


FIGURE3.1: RSTUDIO INTERFACE

For now, all you need to know is that you type R code in the console pane, and press enter to run it. You'll learn more as we go along.

3.3 Tidyvers

You'll also need to install some R packages. An R package is a collection of functions, data, and documentation that extends the capabilities of base R. Using packages is key to the successful use of R. The majority of the packages that you will learn in this book are part of the so-called tidyverse. The packages in the tidyverse share a common philosophy of data and R programming, and are designed to work together naturally.

You can install the complete tidyverse with a single line of code:

```
install.packages("tidyverse")
```

On your own computer, type that line of code in the console, and then press enter to run it. R will download the packages from CRAN and install them on to your computer. If you have problems installing, make sure that you are connected to the internet, and that https://cloud.r-project.org/ isn't blocked by your firewall or proxy.

You will not be able to use the functions, objects, and help files in a package until you load it with library (). Once you have installed a package, you can load it with the library () function:

```
library(tidyverse)
#> Loading tidyverse: ggplot2
#> Loading tidyverse: tibble
#> Loading tidyverse: tidyr
#> Loading tidyverse: readr
#> Loading tidyverse: purrr
#> Loading tidyverse: dplyr
#> Conflicts with tidy packages ----------------------------------------------
#> filter(): dplyr, stats
#> lag():    dplyr, stats
```

This tells you that tidyverse is loading the ggplot2, tibble, tidyr, readr, purrr, and dplyr packages. These are considered to be the core of the tidyverse because you'll use them in almost every analysis.

Packages in the tidyverse change fairly frequently. You can see if updates are available, and optionally install them, by running tidyverse_update().

3.4 Other Packages

There are many other excellent packages that are not part of the tidyverse, because they solve problems in a different domain, or are designed with a different set of underlying principles. This doesn't make them better or worse, just different. In other words, the complement to the tidyverse is not the messy verse, but many other universes of interrelated packages. As you tackle more data science projects with R, you'll learn new packages and new ways of thinking about data.

4

chapter four

project Phases

- Collect data
- Pre-processing(Data cleaning)
- Data transformation
- Data visualization

4.1 Collect Data

When start in the project and search the answers to our questions we must collect the correct and related data.

It is important step in all projects which mean that provide the correct data, get the good answers.

It is very important that the way of read the data. Read as table or read as data frame, etc...

Also it is very important to take care the type of data file excel, csv, rdff, etc...

Resources of Data

We collect data from UK government traffic agency.

4.1.1 Data divided into two files:

1) The Average Annual Daily Flow (ukTrafficAADF.csv). This runs from 2000 – 2016.

2) The accidents data (This had to split into 3 files so it could be handled well in memory) - There are three sheets covering 1.6 million accidents.

The columns are exactly the same, Each file covers a different time

Period (in the file title). The total time period is 2005 - 2014 (2008 is

Missing). (Government, 2017)

4.1.2 Manipulate Data

We combine the accidents files from different period in one file.

4.2 Pre-processing (Data cleaning)

This process is the second step after collect data when working on data. This process mean make data ready for working on it where removing null values duplicated data, etc

Data cleaning, or data preparation, is an essential part of statistical analysis. In fact, in practice it is often more time-consuming than the statistical analysis itself. Data cleaning may profoundly influence the statistical statements based on the data. Typical actions like imputation or outlier handling obviously influence the results of statistical analyses. For this reason, data cleaning should be considered a statistical operation, to be performed in a reproducible manner. The R statistical environment provides a good environment for reproducible data cleaning since all cleaning actions can be scripted and therefore reproduced.

FIGURE4.1: DATA CLEANING STAGES

4.2.1 Problems with Data:

Several issues crop up again and again when preparing data for analysis in R:

1-Loading data into R from databases, spreadsheets, or other formats

2-Shaping data into a form appropriate for analysis

3-Checking variable types: Are the variables of the type that you expect?

4-Managing bad values: What do you do with NaNs or values that are out of range or invalid? How do you find and deal with sentinel values?

5-Dealing with missing values (NA)

6-Anticipating future novel categorical values (values that were not present in training data)

7-Re-encoding categorical values with too many levels: How do you use such variables in an analysis efficiently?

4.2.2 Missing Value:

A missing value, represented by NA in R, is a placeholder for a datum of which the type is known but its value isn't. Therefore, it is impossible to perform statistical analysis on data where one or more values in the data are missing. One may choose to either omit elements from a dataset that contain missing values or to impute a value, but missingness is something to be dealt with prior to any analysis.

How to know Missing Value:

The behavior of R's core functionality is completely consistent with the idea that the analyst must decide what to do with missing data. A common choice, namely "leave out records with missing data" is supported by many base functions through the na.rm option.

Functions such as sum (), prod (), quantile(), and so on all have this option.

Functions implementing bivariate statistics such as cor () and cov() offer options to include complete or pairwise complete values.

```{r}
sum(is.na(AllAccidents))
```

```
[1] 1470399
```

To Remove Missing Value:

is.na ():

Besides the is.na () function, that was already mentioned previously, R comes with a few other functions facilitating NA handling. The complete.cases () function detects rows in a data.frame that do not contain any missing value.

The function na.omit () returns the object with listwise deletion of missing values. (RDocumentation, 2017)

```{r}
na.omit(AllAccidents)
```

0 rows | 1-5 of 33 columns

4.2.3 Duplicated Data:

Duplicated data mean that there are many values is repeated

To know Duplicated Values:

By using duplicated () function and using sum () function to know number of duplicated data at your dataset:

```
```{r}
sum(duplicated(AllAccidents))
```
```

```
[1] 0
```

To Remove Duplicated Data:

We can remove duplicated data by using function called unique() which remove all duplicated in your data.

4.2.4 Convert Empty Value To NULL:

Change it in two steps.

1. Convert empty value with NULL.
2. fill Null value with last observation carried forward

```
# empty value
table(allacc$Junction_Control[allacc$Junction_Control==""])
```

```
##
##                               Authorised person Automatic traffic signal
##                         0                       0                       0
##  Giveway or uncontrolled                Stop Sign
##                         0                       0
```

```
allacc<-allacc%>%
 fill(Junction_Control)


# remove not completed value
allacc<-allacc[complete.cases(allacc),]

head(allacc$Junction_Control,n=15)
```

```
##  [1] Automatic traffic signal Automatic traffic signal
##  [3] Automatic traffic signal Automatic traffic signal
##  [5] Automatic traffic signal Giveway or uncontrolled
##  [7] Giveway or uncontrolled  Automatic traffic signal
##  [9] Giveway or uncontrolled  Automatic traffic signal
## [11] Automatic traffic signal Giveway or uncontrolled
## [13] Giveway or uncontrolled  Giveway or uncontrolled
## [15] Giveway or uncontrolled
## 5 Levels:  Authorised person ... Stop Sign
```

Filter ():

Filtering data is one of the very basic operation when you work with data. You want to remove a part of the data that is invalid or simply you're not interested in. Or, you want to zero in on a particular part of the data you want to know more about. Of course, dplyr has filter () function to do such filtering, but there is even more. With dplyr you can do the kind of filtering, which could be hard to perform or complicated to construct with tools like SQL and traditional BI tools, in such a simple and more intuitive way. (RDocumentation, 2017)

```
ukTrafficAADF<-filter(ukTrafficAADF,!(ukTrafficAADF$AADFYear%in%c(2000:2004)))
table(ukTrafficAADF$AADFYear)
```

```
##
##  2005  2006  2007  2008  2009  2010  2011  2012  2013  2014  2015  2016
## 16306 16366 16363 16338 16379 16358 16344 16445 16467 16500 16504 14169
```

using filter function remove 2008 traffic data.

```
ukTrafficAADF<-filter(ukTrafficAADF,!(ukTrafficAADF$AADFYear==2008))
table(ukTrafficAADF$AADFYear)
```

```
##
##  2005  2006  2007  2009  2010  2011  2012  2013  2014  2015  2016
## 16306 16366 16363 16379 16358 16344 16445 16467 16500 16504 14169
```

Gather ():

Gather () does the reverse of spread (). Gather () collects a set of column names and places them into a single "key" column. It also collects the cells of those columns and places them into a single value column.

To use gather (), pass it the name of a data frame to reshape. Then pass gather() a character string to use for the name of the "key" column that it will make, as well as a character string to use as the name of the value column that it will make. Finally, specify which columns gather () should collapse into the key value pair (here with integer notation).

Gather () returns a copy of the data frame with the specified columns removed. To this data frame, gather () has added two new columns: a "key" column that contains the former column names of the removed columns, and a value column that contains the former values of the removed columns. gather () repeats each of the former column names (as well as each of the original columns) to maintain each combination of values that appeared in the original data set. gather() uses the first string that you supplied as the name of the new "key" column, and it uses the second string as the name of the new value column. (Teetor, 2011)

Separate ():

Separate () turns a single character column into multiple columns by splitting the values of the column wherever a separator character appears.

To use separate () pass separate the name of a data frame to reshape and the name of a column to separate. Also give separate() an into argument, which should be a vector of character strings to use as new column names. separate () will return a copy of the data frame with the column removed. The previous values of the column will be split across several columns, one for each name in into.

By default, separate() will split values wherever a non-alphanumeric character appears. Non-alphanumeric characters are characters that are neither a number nor a letter. For example, in the code above, separate() split the values of rate at the forward slash characters.

Unit ():

Unite () does the opposite of separate (): it combines multiple columns into a single column.

Give unite () the name of the data frame to reshape, the name of the new column to create (as a character string), and the names of the columns to unite. unite () will place an underscore (_) between values from separate columns. If you would like to use a different separator, or no separator at all, pass the separator as a character string to sep.

unite() returns a copy of the data frame that includes the new column, but not the columns used to build the new column. If you would like to retain these columns, add the argument remove = FALSE

4.3 Data Transformation

This phase we are going to edit in data to facilitate our work on it.

Ex: the variable of type integer and contain date, so we will transform the data from integer type into date type.

And we can find data in char type contain number values so we need to change it to integer values to can do mathematics operations.

Adding New Variables

This process we do it when need to combine two variables or more together and make operation in the result, so we need to save the result in new variable (column) and save the original data as it was.

Editing Current Variables

We found some column have the data which need to change type of columns or multiply in factor before use it.

Filtering and Selecting Variables

The data contain many variables and rows. So when need to get the results which answer the question, we must select the columns which help to achieve the columns and filter rows when need specific conditions in results.

Editing Structure of Variables

We go to change the structure of data when variables contain values cannot use it in its current structure or type, so we go to change the type of data for facilitating our process.

4.4 Data Visualization

Visualize this basic idea of analysis, this mean that make graphs and insights make any one understand the data easily. Making good visualization must choose the right type of chart for your data.

Data visualization is viewed by many disciplines as a modern equivalent of visual communication. It involves the creation and study of the visual representation of data, meaning "information that has been abstracted in some schematic form, including attributes or variables for the units of information".

A primary goal of data visualization is to communicate information clearly and efficiently via statistical graphics, plots and information graphics. Numerical data may be encoded using dots, lines, or bars, to visually communicate a quantitative message. Effective visualization helps users analyze and reason about data and evidence. It makes complex data more accessible, understandable and usable. Users may have particular analytical tasks, such as making comparisons or understanding causality, and the design principle of the graphic (i.e., showing comparisons or showing causality) follows the task. Tables are generally used where users will look up a specific measurement, while charts of various types are used to show patterns or relationships in the data for one or more variables.

Data visualization is both an art and a science. It is viewed as a branch of descriptive statistics by some, but also as a grounded theory development tool by others. Increased amounts of data created by Internet activity and an expanding number of sensors in the environment are referred to as "big data" or Internet of things. Processing, analyzing and communicating this data present ethical and analytical challenges for data visualization.

The field of data science and practitioners called data scientists help address this challenge. (Wickham, 2017)

4.4.1Prerequisites:

GGplot2:

GGplot2 is a plotting system for R, based on the grammar of graphics, which tries to take the good parts of base and lattice graphics and none of the bad parts. It takes care of many of the fiddly details that make plotting a hassle (like drawing legends) as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics.

GGplot2:

Provides two ways to produce plot objects:

qplot ():

Uses some concepts of The Grammar of Graphics, but doesn't provide full capability and designed to be very similar to plot() and simple to use

May make it easy to produce basic graphs but may delay understanding philosophy of ggplot2

GGplot ():

Provides fuller implementation of The Grammar of Graphics may have steeper learning curve but allows much more flexibility when building graphs

4.4.2 Grammar Defines Components of Graphics:

Data: in ggplot2, data must be stored as an R data frame

Coordinate system: describes 2-D space that data is projected onto - for example, Cartesian coordinates, polar coordinates, map projections.

Geoms: describe type of geometric objects that represent data - for example, points, lines, polygons,

Aesthetics: describe visual characteristics that represent data - for example, position, size, color, shape, transparency, fill

Scales: for each aesthetic, describe how visual characteristic is converted to display values - for example, log scales, color scales, size scales, shape scales.

Stats: describe statistical transformations that typically summarize data - for example, counts, means, medians, regression lines, ...

Facets: describe how data is split into subsets and displayed as multiple small graphs

To install ggplot2 ():

You can install ggplot2 () by using this command

```
install.packages("ggplot2")
library(ggplot2)
```

Plot geoms:

- Geom = "point" draws points to produce a scaterplot. This is the default when you supply both x and y arguments to qplot ().
- geom = "smooth" fits a smoother to the data and displays the smooth and its standard error
- geom = "boxplot" produces a box and whisker plot to summarise the distribu/on of a set of points

- Geom = "path" and geom = "line" draw lines between the data points.
- For con/nuous variables, geom = "histogram" draws a histogram, geom = "freqpoly" a frequency polygon, and geom = "density" creates a density plot
- For discrete variables, geom = "bar" makes a barchart

Plot generation process:

•Each square represents a layer, and this schematic represents a plot with three layers and three panels.

• All steps work by transforming individual data frames, except for training scales which doesn't affect the data frame and operates across all datasets simultaneously.

4.4.3 Basic Graph Functions:

1-Creating bar charts:

A bar chart is the simplest chart that displays the frequency of a categorical variable or the summary statistics of a numeric variable over the category of other variables.

To do it:

The standard code that produces a bar chart is as follows:

```
ggplot(data= diamonds, aes(x=cut, y=price,fill=cut)) + geom_
bar(stat="identity")+theme_bw()
```

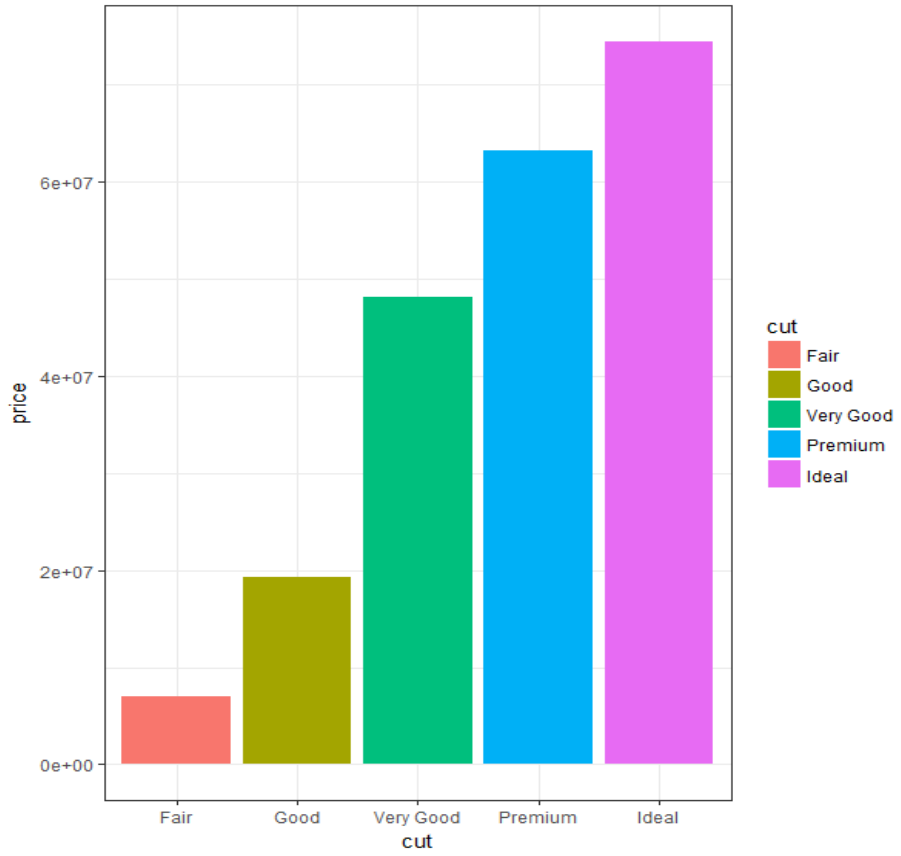


figure4.2: sample of barplot graph function

2-Visualizing the Density of a numeric Variable:

The density plot is mostly used to compare distributions of two numeric variables or a single numeric variable over the category of other variables. In this recipe, we will produce a density plot using the ggplot2 library.

To do:

To create a density plot, the basic code is as follows:

```
ggplot(data=diamonds,aes(x=cut,color=cut))+geom_density()
```

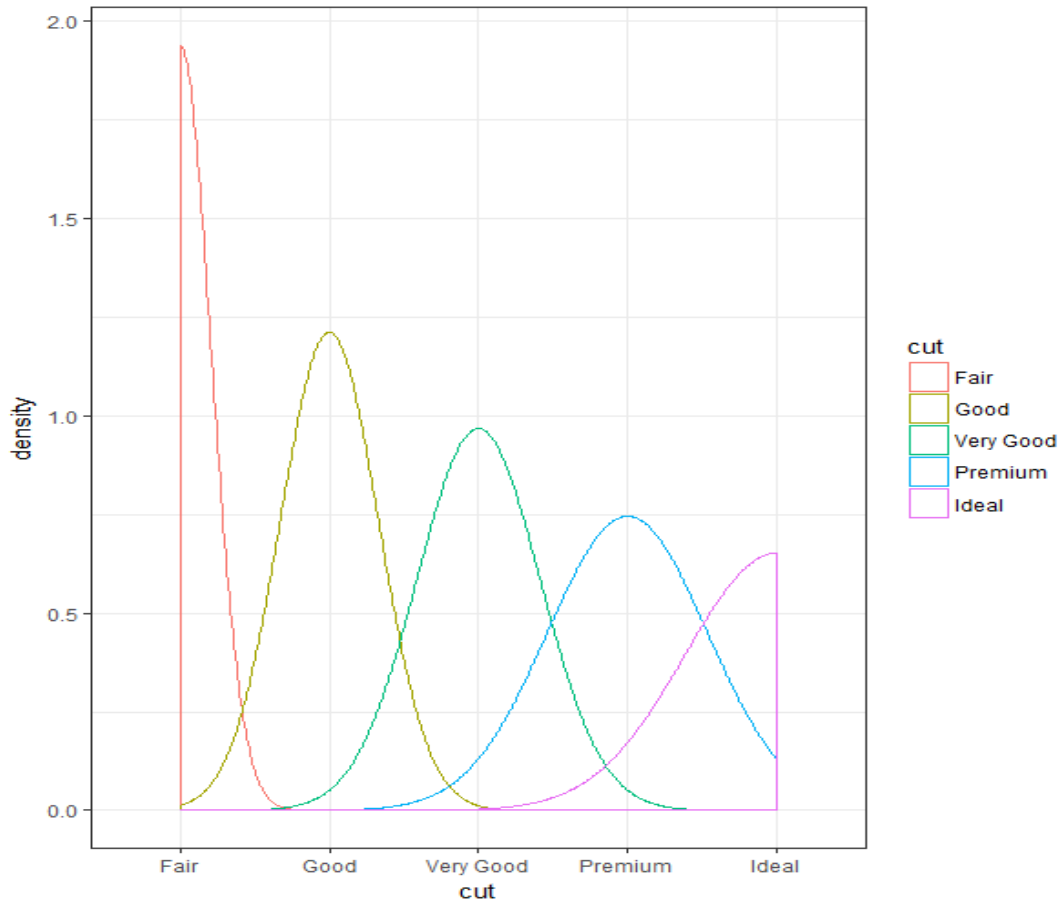


FIGURE4.3: SAMPLE OF DENSITY GRAPH FUNCTION

3-Creating a box plot:

A box plot is another important graph that summarizes the data along with the distribution. In this recipe, we will produce a box plot to visualize the data summary with the distribution using the ggplot2 implementation.

To do:

The primary code to produce a boxplot is as follows:

```
ggplot(data=diamonds,aes(y=cut,x=price,fill=cut))+geom_boxplot()+them
e_bw()
```

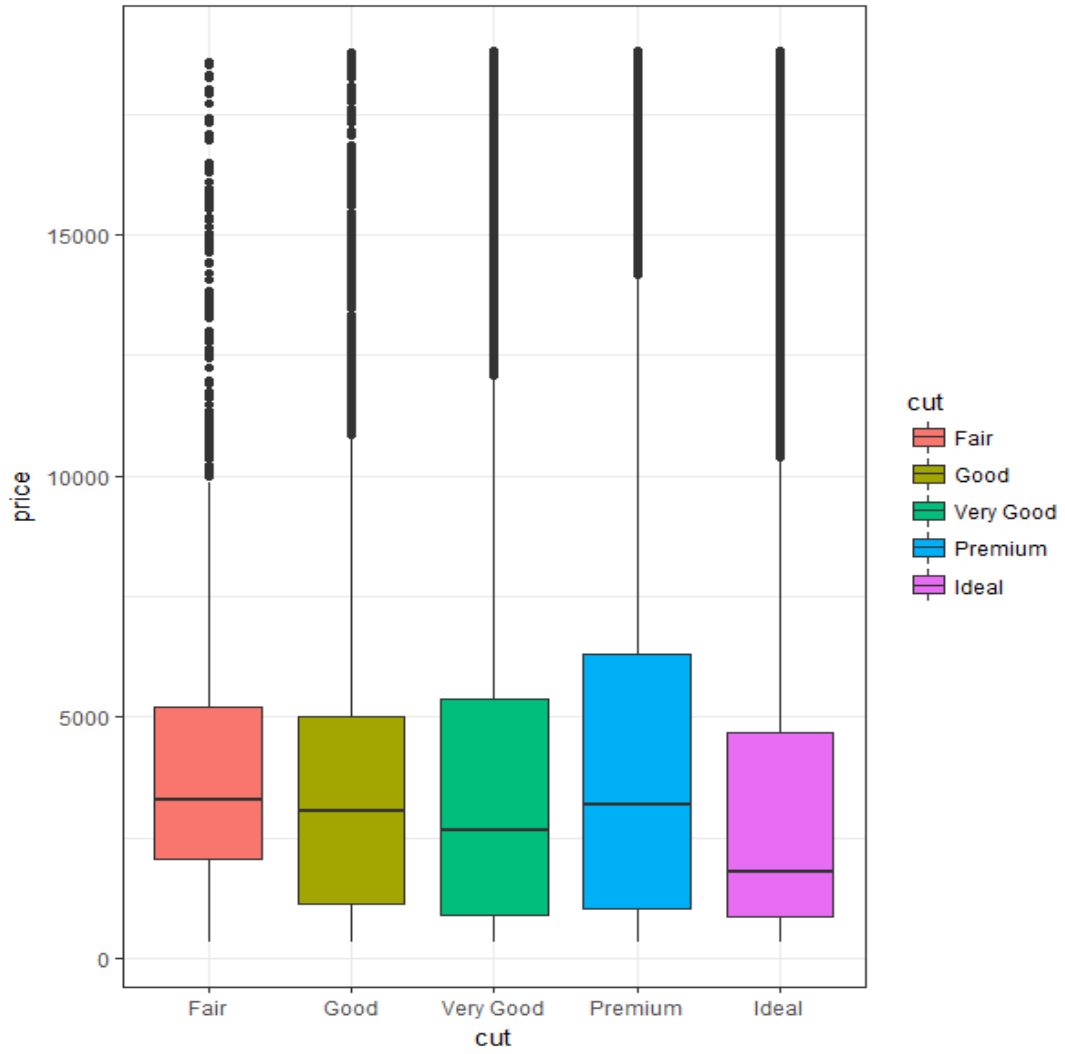


FIGURE4 4: SAMPLE OF BOXPLOT GRAPH FUNCTION

The main visualization is produced by the geom_boxplot() part of the function. The ggplot function takes the argument of the data frame that we want to use, along with the x- and y-axis variable names. The geom part maps the data and produces the actual visualization

4-Created a layered plot with a scatter plot and fitted line:

To visualize a relation between two numeric variables, we usually create a scatter plot. We also add a fitted line or smooth curve to the scatter plot, which represents the majority of the data points. In this recipe, we will see how we can produce a scatter plot and then add a layer of fitted line

along with a linear smooth and curved line. The curved line could be the lowess or local regression, but here, is loess the default.

To do:

Perform the following steps:

1. We can easily produce the scatter plot with the ggplot function along with the geom

point options, as follows:

```
Sctrplot <- ggplot(data=diamonds,aes(x=depth,y=price))+geom_point()
```

2. We will add a layer of fitted linear line to the scatter plot object, as follows:

```
Sctrplot <- Sctrplot + geom_smooth(method="lm",col="red")
```

3. Now, we will add another layer to the curved line, as follows:

```
Sctrplot + geom_smooth(col="green")
```

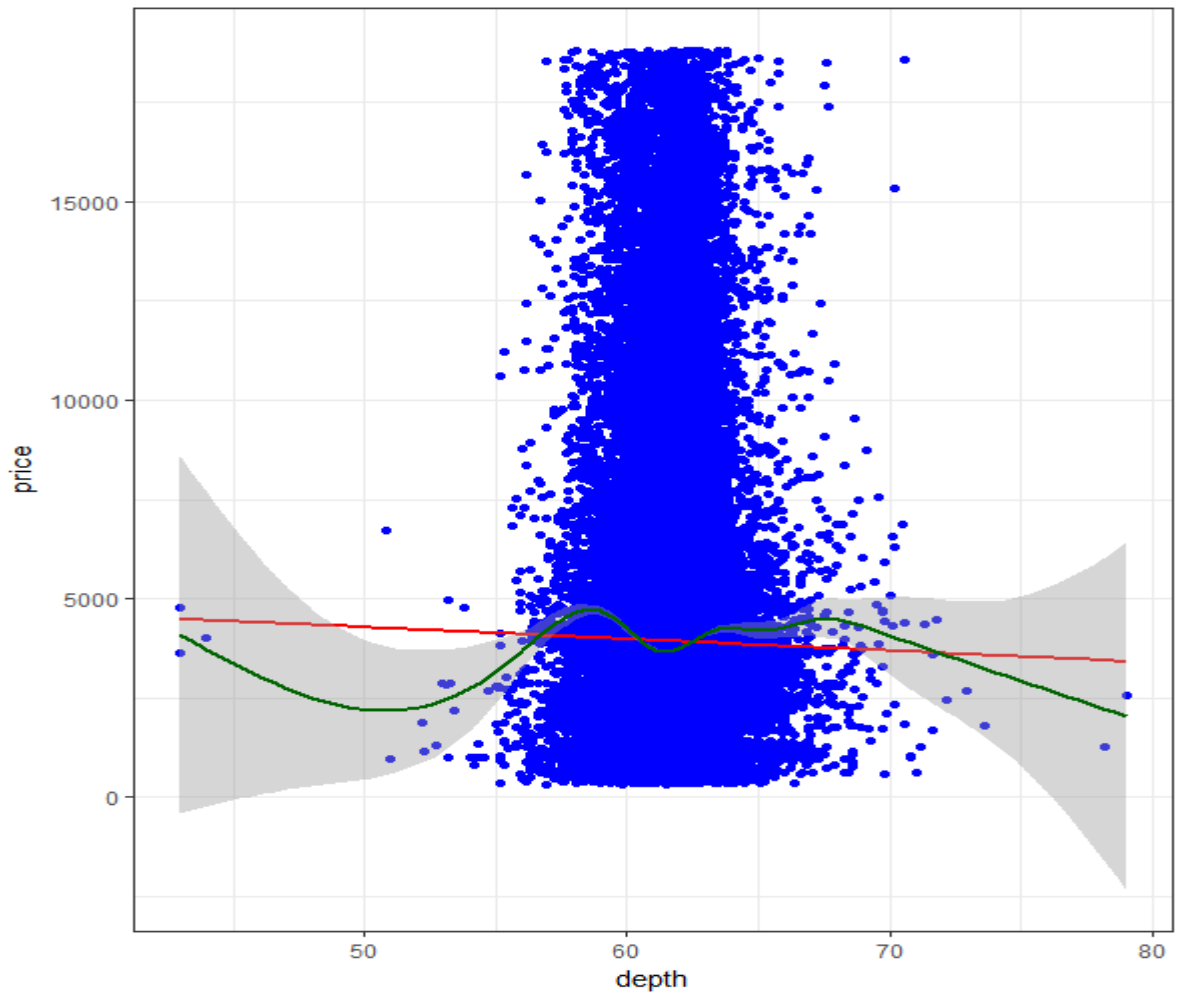


FIGURE4 5: SAMPLE OF SCATERPLOT GRAPH FUNCTION

Firstly, we created a plot object. Then, we added layers along with different visualizations one by one. The original plot was a scatter plot of two numeric variables. Then we added least square fitted line with the default 95 percent prediction interval. We then added another curved line with the default 95 percent confidence interval. To distinguish the two different lines, we used different colors.

5-Creating a line chart:

Line charts are sometimes useful in order to view the pattern of a single variable against its index. Also, in some cases, bivariate connected lines are also useful. In this recipe, we will draw a line chart using ggplot2 with geom_line.

To do:

To produce a connected line of a single numeric variable against its observation index, we will use the following code:

connected line

ggplot(data=trainingdata,aes(x=1:nrow(trainingdata),y=accident_Rate))+geom_line()

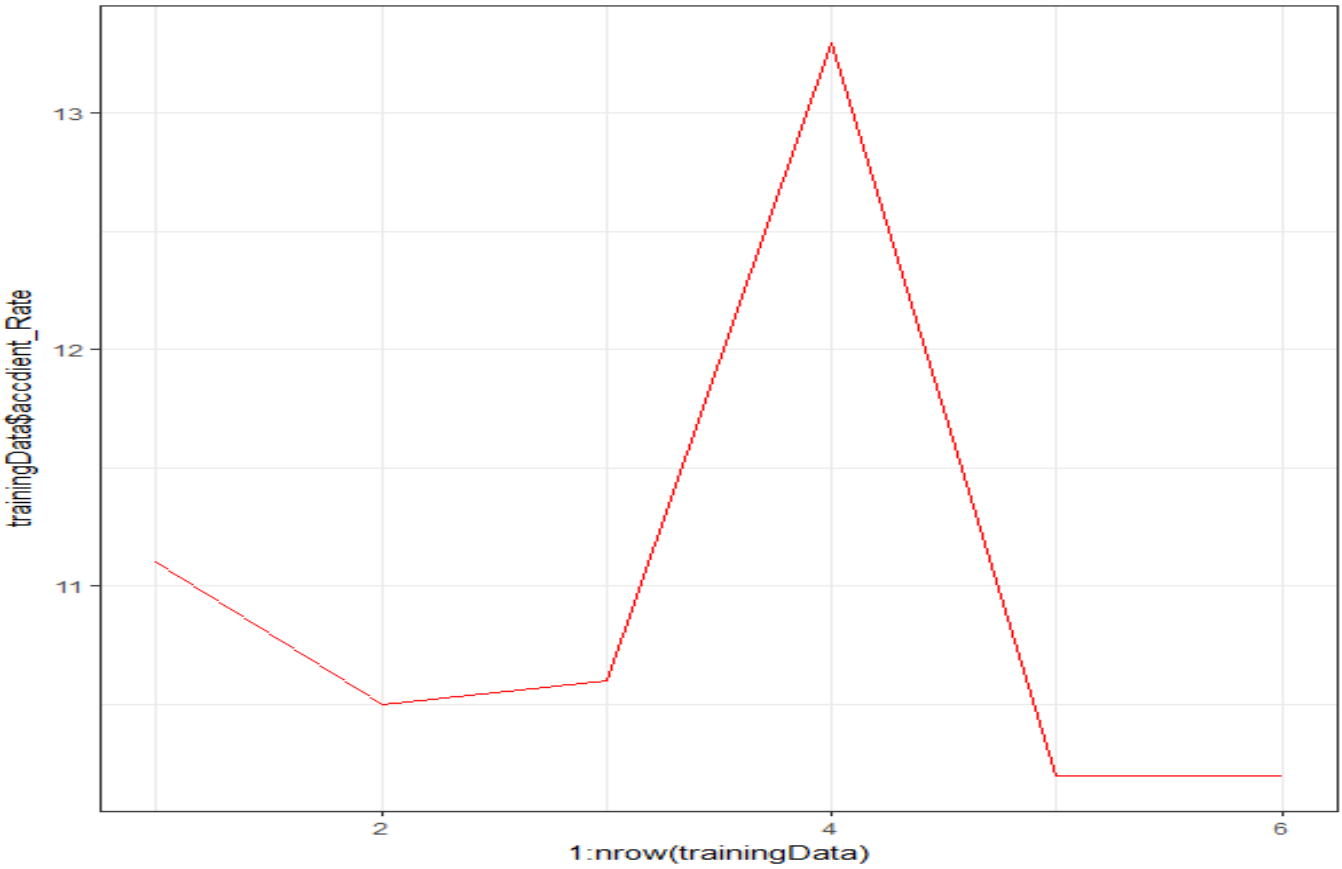


FIGURE4.6 : SAMPLE OF CONNECTED LINE GRAPH FUNCTION

The single variable's connected line looks like a time series plot, but the difference is that this plot does not have the time axis. The x argument

within aes() specifies the values of the x-axis. In this case, the value of x-axis is just the sequence number starting from 1 to the number of observations in the dataset. So, the aes() function internally produces

coordinates in the x-y plane with x with the index value and y, takes the value from the disA variable, and then connects the points in order to produce the line.

6- Graph annotation with ggplot2:

To produce publication-quality data visualization, we often need to annotate the graph with various texts, symbols, or even shapes. In this recipe, we will learn how we can easily annotate an existing graph.

To do:

In this recipe, we will execute the following annotation on an existing scatter plot.

So, the whole procedure will be as follows:

1. Create a scatter plot.

2. Add customized text within the plot.

3. Highlight a certain region to indicate extreme values.

4. Draw a line segment with an arrow within the scatter plot to indicate a single extreme observation.

7- Plotting multiple groups with facets with ggplot2:

Faceting:

Facets divide a plot into subplots based on the values of one or more discrete variables.

Facet_grid():

Form a matrix of panels defined by row and column facetting variables. It is most useful when you have two discrete variables, and all combinations of the variables exist in the data.

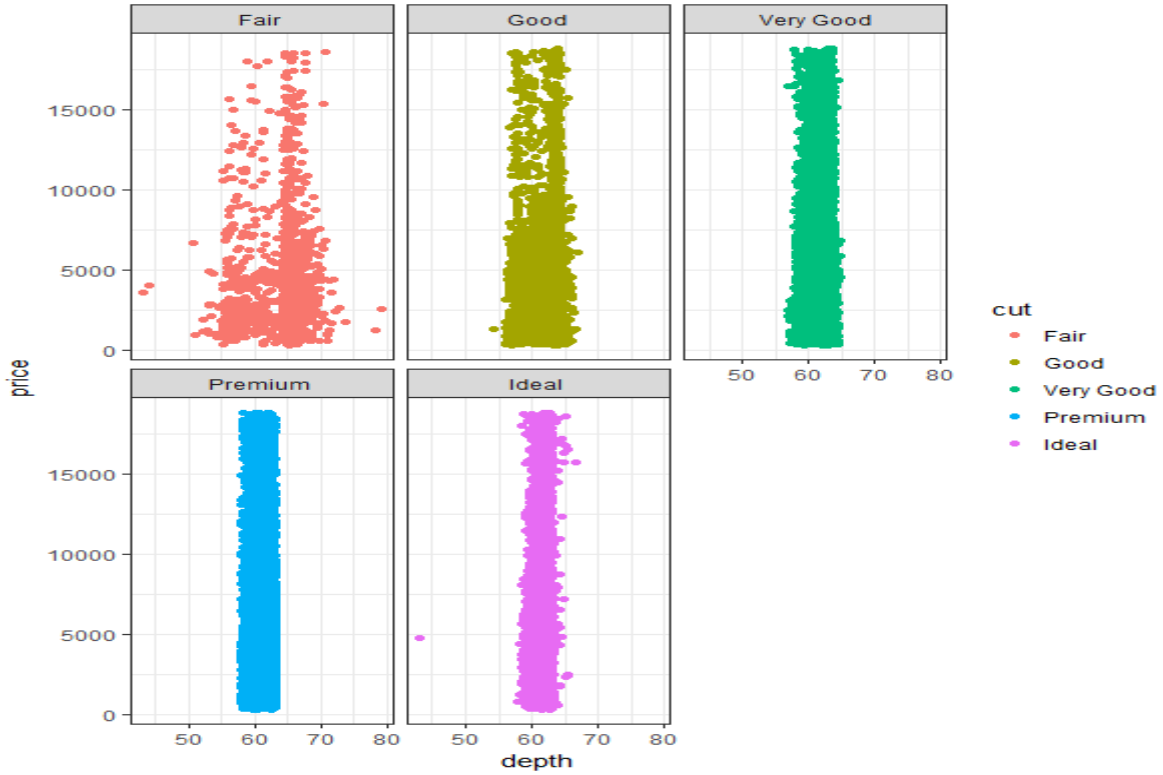


FIGURE4.7: SAMPLE OF FACET_GRID

5

chapter five

Machine Learning

- Regression analysis&time series analysis.
- Time series and forecasting.
- Simple linear Regression(SLR).
- Prediction accident Rate for each month.

5.1-Regression analysis and time series analysis.

Objectives:

- predict of accident rate
- Forcasting for future accident rate from 2015 to 2018.

Regression analysis: is a powerful statistical method that allows you to examine the influence of one or more independent variables on a dependent variable.

- Dependent variable: the variable we wish to predict
  Or explain.

- Independent variable: the variable used to explain the dependent variable.

5.2-Time Series and Forecasting

Time series Analysis: Series of Data points indexed in time order.

Time series forecasting: is the use of model to predict future value based on previously observed values.

❖ Components of Time Series Data.

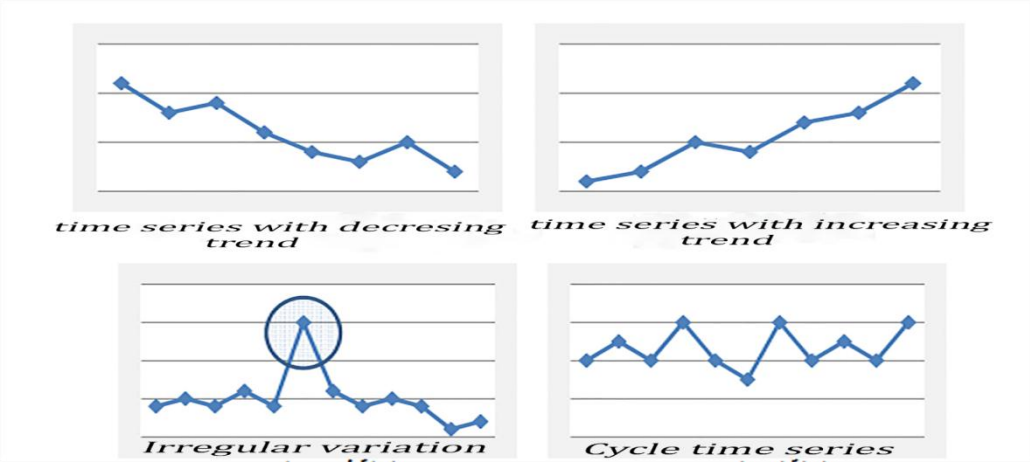


FIGURE5.1: COMPONENTS OF TIME SERIES DATA

5.3-Simple Linear Regression Model

Relationship between dependent variable and independent variable is described by a linear function.

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

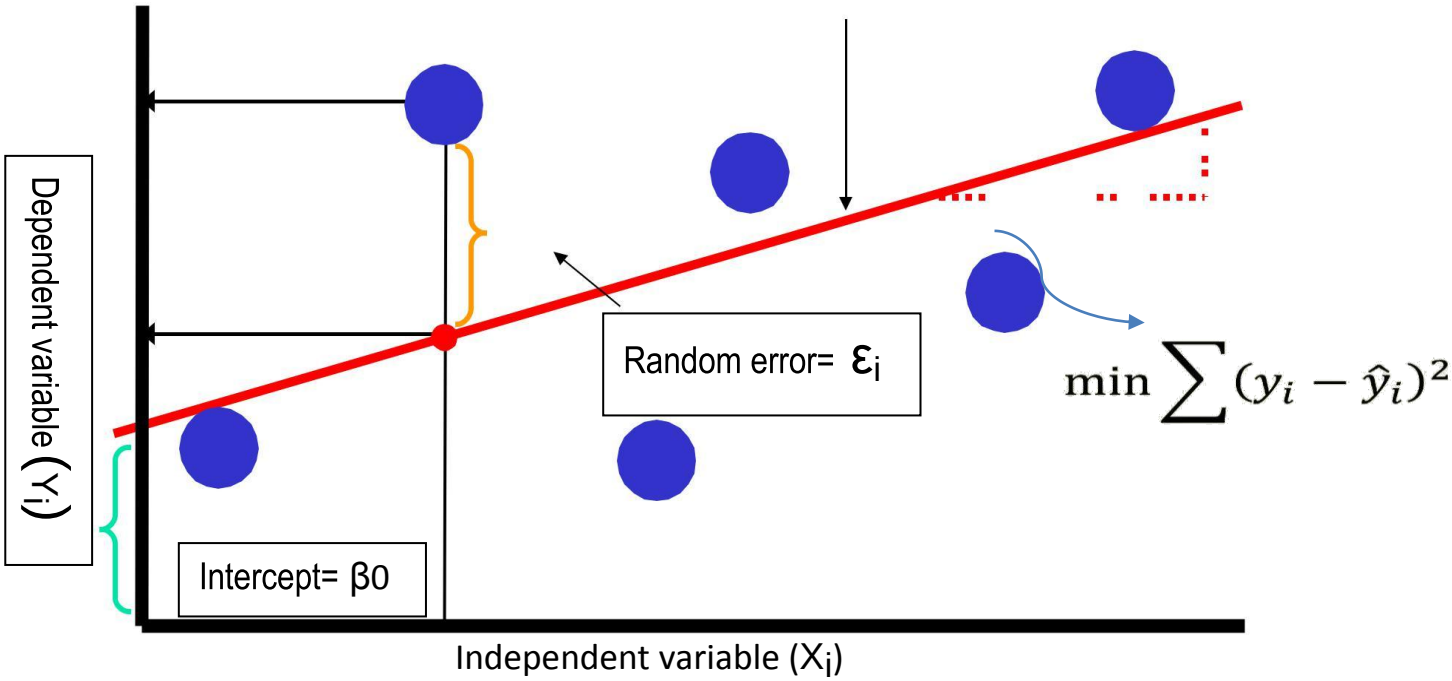


FIGURE5.2:RELATIONSHIP BETWEEN DEPENDENT VARIABLE AND INDEPENDENT VARIABLE

β1= $\frac{n\sum xy - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2}$ Slope

β0= $\frac{\sum y - b\sum x}{n}$ Intercept

5.3.1-Linear Model:

Model: From the description of the problem, it says that this a time series data where Accidents rate depends on the year.

Thus dependent variable(y) is accidents rate and independent variable is Year (x). We wish to fit a liner model Y = α + βx

Model :accidentas Rate=α + β*Year

Step II:

The following scatter diagram shows that 1,there is a inverse relationship between x and y, that is as Year increase, accidentas Rate decrease decreases. 2. We see a distinct linear trend among the data points supporting our model in step I.

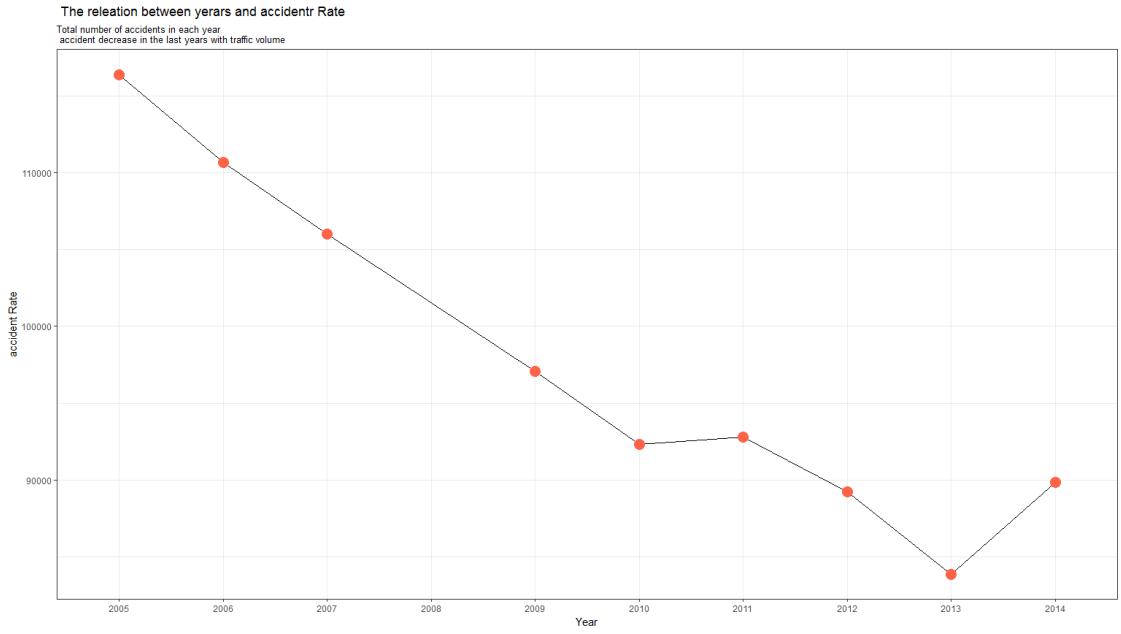
-The releation between years and number of accidents.

```
# plot2

ggplot(data=x1,aes(x=Year,y=accidents),na.rm=F)+
 geom_line()+
 scale_x_discrete(limits = c(2005:2014))+


 labs(
  title=" The releation between yerars and accidentr Rate",
  subtitle="Total number of accidents in each year\n accident decrease in the
last years with traffic volume",
  x="Year",
  y="accident Rate"

 )+theme_bw()+geom_point(size=5,color="Tomato")#+
```



```
 #geom_text(aes(label=paste0(accdient_Rate,"%")),vjust=-
0.9,size=4,color="ForestGreen")
```

This is a good thing, because, one of the underlying assumptions in linear regression is that the relationship between the response and predictor variables is linear.

5.3.2 -How to know if the model is best fit for your data?

We estimate each the follwing:

- R-squared & Adjusted R-squared: Higher the better (> 0.70)

R-squared: 0.911, Adjusted R-squared: 0.8983

- F-Statistic: Higher the better.
- Std. Error: closer to zero.
- t-statistic: should be greater 1.96 for p-value to be less than 0.05.
- AIC $BIC: Lower the better.
- Min_Max Accuracy => mean(min(actual, predicted)/max(actual, predicted)): Higher the better.

the model with the lowest AIC and BIC score is preferred.

```
AIC(linearMod)  # AIC => The Akaike's information criterion - AIC
```

```
## [1] 16.04577
```

```
BIC(linearMod)  # BIC =>the Bayesian information criterion - BIC
```

```
## [1] 16.63744
```

5.3.3-Data prepation:

1-Group accidents by year.

```
x<-group_by(accidents,Year)
```

2-Summarise by years to find total number of accidents in each year.

```
x1<- dplyr::summarize(x,accidents = n())
x1<-mutate(x1,trend=1:length(x1$Year)) #
head(x1)
```

```
## # A tibble: 6 x 3
##    Year accidents trend
##   <int>     <int> <int>
## 1  2005    116387     1
## 2  2006    110655     2
## 3  2007    105988     3
## 4  2009     97061     4
## 5  2010     92313     5
## 6  2011     92787     6
```

3-Compute percent % for accidents in each year.

```
# compute percentage.
x1<-mutate(x1,accdient_Rate=round(accidents/nrow(x)*100,1)
)

# show summarize:


accident<-x1
head(accident)
```

```
## # A tibble: 6 x 4
##    Year accidents trend accdient_Rate
##   <int>     <int> <int>         <dbl>
## 1  2005    116387     1          13.3
## 2  2006    110655     2          12.6
## 3  2007    105988     3          12.1
## 4  2009     97061     4          11.1
## 5  2010     92313     5          10.5
## 6  2011     92787     6          10.6
```

5.3.4-Predicting Linear Models:

The preferred practice is to split our dataset into a 80:20 sample (training:test), then, build the model on the 80% sample and then use the model thus built to predict the dependent variable on test data.

Step 1: Create the training (development) and test (validation) data samples from original data.

```
# Create Training and Test data -
set.seed(80)  # setting seed to reproduce results of random sampling
trainingRowIndex <- sample(1:nrow(accident), 0.75*nrow(accident))  # row
```

```
indices for training data
trainingData <- accident[trainingRowIndex, ]  # model training data
testData  <- accident[-trainingRowIndex, ]   # test data
testData
```

```
## # A tibble: 3 x 4
##    Year accidents trend accdient_Rate
##   <int>     <int> <int>         <dbl>
## 1  2006    110655     2          12.6
## 2  2007    105988     3          12.1
## 3  2013     83852     8           9.50
```

```
trainingData
```

```
## # A tibble: 6 x 4
##    Year accidents trend accdient_Rate
##   <int>     <int> <int>         <dbl>
## 1  2009     97061     4          11.1
## 2  2010     92313     5          10.5
## 3  2011     92787     6          10.6
## 4  2005    116387     1          13.3
## 5  2014     89855     9          10.2
## 6  2012     89241     7          10.2
```

Step 2: Develop the model on the training data and use it to predict the distance on test data.

```
# Build the model on training data -
```

```
lmMod <- lm(accdient_Rate ~ trend, data=trainingData)  # build the model
distPred <- predict(lmMod, testData)  # predict distance
distPred
```

```
##       1       2       3
## 12.2750 11.8875  9.9500
```

1-vizualize test and predicted data.

The releation between actual accident Rate and predicated accident Rate.

```
# Visualising the Test set results
library(ggplot2)
ggplot() +
  geom_point(aes(x = testData$Year, y = testData$accdient_Rate),
             colour = 'blue',size=4)+geom_line(aes(x = testData$Year, y =
testData$accdient_Rate),
                                          colour = 'blue')+
  geom_line(aes(x = testData$Year, y = distPred),colour = 'red')
+geom_point(aes(x = testData$Year, y =       distPred),colour = 'red',size=4) +
  ggtitle('accidentRate vs Year (Test set)') +scale_x_discrete(limits
=c(2005:2014))+
  scale_y_discrete(limits=seq(100))+
```

```
  xlab('Year') +
  ylab('accient Rate')+ggtitle("Note: redline:predicted accident Rate\n blue
actual value.")+
 geom_text(aes(x = testData$Year, y =
testData$accdient_Rate,label=paste0(testData$accdient_Rate,"%")),vjust=-
0.8,size=5,color="black")+
 theme_bw()+


 geom_text(aes(x = testData$Year, y=
distPred,label=paste0(round(distPred,1),"%")),vjust=-
0.9,size=5,color="DarkGreen")
```

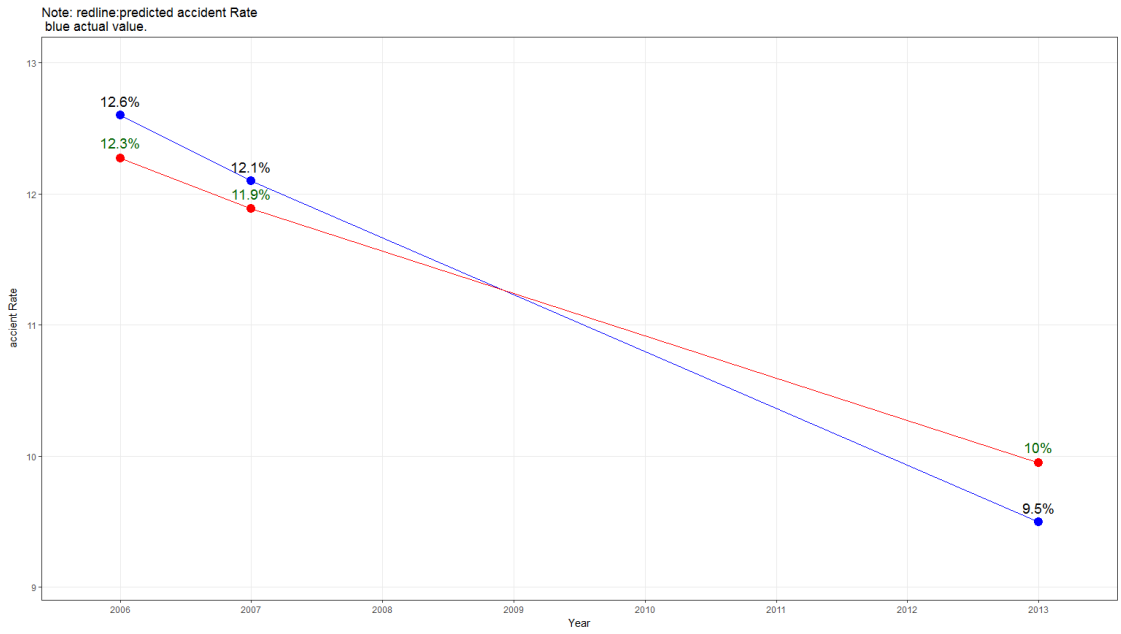


FIGURE5.3:THE RELATION BETWEEN ACTUAL ACCIDENT RATE AND PREDICATED ACCIDENT RATE

Step 3: Review diagnostic measures.

```
summary (lmMod)  # model summary
```

```
##
## Call:
## lm(formula = accdient_Rate ~ trend, data = trainingData)
##
## Residuals:
##       1       2       3       4       5       6
## -0.4000 -0.6125 -0.1250  0.6375  0.6375 -0.1375
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.05000    0.56651  23.036  2.1e-05 ***
## trend       -0.38750    0.09622  -4.027   0.0158 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5879 on 4 degrees of freedom
## Multiple R-squared:  0.8022, Adjusted R-squared:  0.7527
## F-statistic: 16.22 on 1 and 4 DF,  p-value: 0.01577
```

```
AIC (lmMod)
```

```
## [1] 14.22007
```

```
BIC(lmMod)
```

```
## [1] 13.59534
```

Step 4: Calculate prediction accuracy and error rates:

A simple correlation between the actuals and predicted values can be used as a form of accuracy measure. A higher correlation accuracy implies that the actuals and predicted values have similar directional movement, i.e. when the actuals values increase the predicteds also increase and vice-versa.

Now lets calculate the Min Max accuracy and MAPE:

MinMaxAccuracy=mean(min(actuals,predicteds)max(actuals,predicteds))

MeanAbsolutePercentageError (MAPE)=mean(abs(predicteds−actuals)actuals)

```
actuals_preds <- data.frame(cbind(actuals=testData$accdient_Rate,
predicteds=distPred))  # make actuals_predicteds dataframe.
correlation_accuracy <- cor(actuals_preds)  #
actuals_preds
```

```
##   actuals predicteds
## 1    12.6    12.2750
## 2    12.1    11.8875
## 3     9.5     9.9500
```

calculate The model Accuracy.

```
min_max_accuracy <- mean(apply(actuals_preds, 1, min) / apply(actuals_preds, 1,
max))
```

```
min_max_accuracy # accuracy is 0.95%
```

```
## [1] 0.9704727
```

```
#Mean Absolute Percentage Error (MAPE)
mape <- mean(abs((actuals_preds$predicteds -
actuals_preds$actuals))/actuals_preds$actuals)*100
  mape
```

```
## [1] 3.024135
```

```
  #Mean Absolute Percentage Error (MAPE) is 4.97%

  acc_erro<-
data.frame(type=c('Accuracy',"Error"),value=c(min_max_accuracy*100,mape))


ggplot(data=acc_erro)+geom_bar(aes(x=type,y=value,fill=type),stat='identity')+
   geom_text(aes(x = type, y= value,label=paste0(round(value,1),"%")),vjust=-
0.9,size=3,color="DarkGreen")+theme_bw()
```

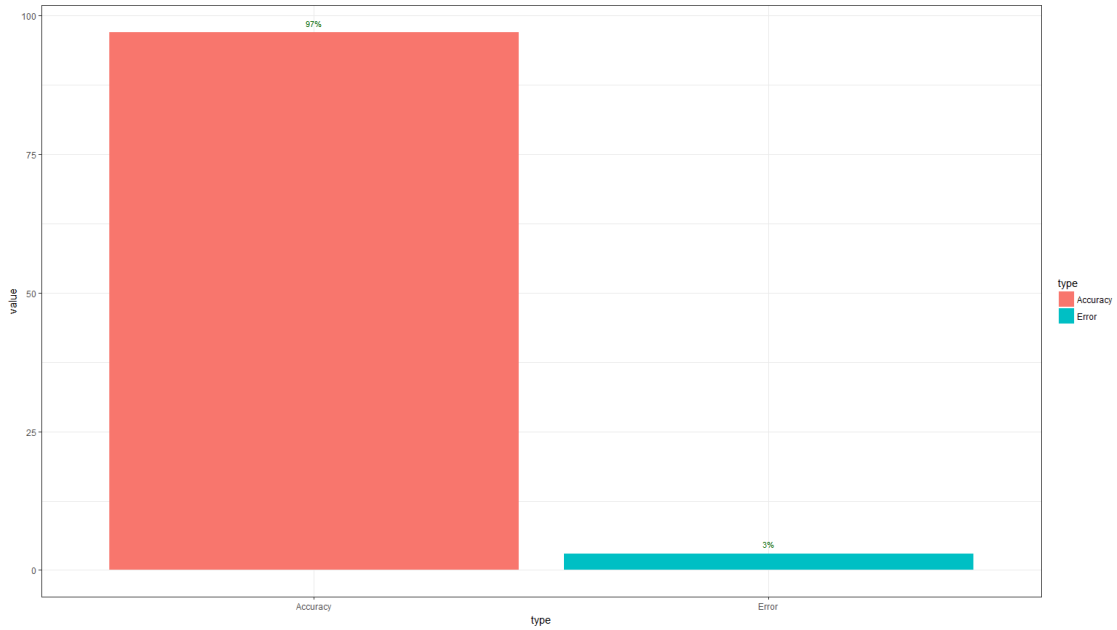


FIGURE5.4: MODEL ACCURACY AND ERROR RATE

`

5.3.5- k-Fold Cross validation:

Suppose, the model predicts satisfactorily on the 20% split (test data), is that enough to believe that your model will perform equally well all the time? It is important to rigorously test the model's performance as much as possible. One way is to ensure that the model equation you have will perform well, when it is 'built' on a different subset of training data and predicted on the remaining data.

How to do this is? Split your data into ‘k’ mutually exclusive random sample portions. Keeping each portion as test data, we build the model on the remaining (k-1 portion) data and calculate the mean squared error of the predictions. This is done for each of the ‘k’ random sample portions. Then finally, the average of these mean squared errors (for ‘k’ portions) is computed. We can use this metric to compare different linear models.

By doing this, we need to check two things:

If the model’s prediction accuracy isn’t varying too much for any one particular sample, and If the lines of best fit don’t vary too much with respect the the slope and level. In other words, they should be parallel and as close to each other as possible. You can find a more detailed explanation for interpreting the cross validation charts when you learn about advanced linear model building.

```
bais<-data.frame(value=c(mape*10,2.15*10),type=c('actual Model','predictor
Models'))

ggplot(data=bais)+geom_bar(aes(x=type,y=value,fill=type),stat='identity')+
   geom_text(aes(x = type, y= value,label=paste0(round(value,1),"%")),vjust=-
0.9,size=3,color="DarkGreen")+theme_bw()
```

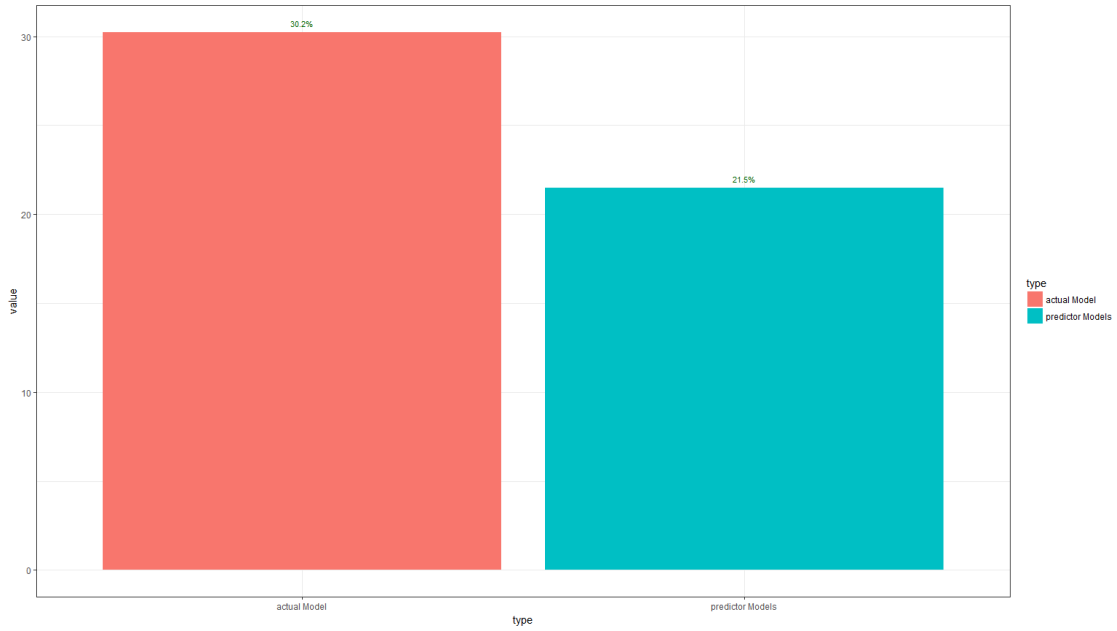


FIGURE5.5: BAISE TRAINING DATA

prediction of accident Rate for new Data 2015 to 2018 that not involved in The model.

```
#predited some point
mydata<-data.frame(trend=c(11,12,13,14))
predict_2008<-predict(lmMod, mydata,interval="confidence" ,level=0.95)  #
predict accident Rate.
predict_2008
```

```
##      fit      lwr      upr
## 1 8.7875 7.133515 10.44149
## 2 8.4000 6.498464 10.30154
## 3 8.0125 5.858727 10.16627
## 4 7.6250 5.215774 10.03423
```

```
mydata<-data.frame(Year=c(2008,2015,2016,2017,2018,2019,2020))
predict_2008<-predict(lmMod, mydata,interval="confidence" ,level=0.95)  #
predict accident Rate.


 ggplot()+

geom_point(aes(x=mydata$Year,y=predict_2008[ ,'fit']),color='darkgreen',size=4)
+

geom_smooth(aes(x=mydata$Year,y=predict_2008[ ,'fit']),linetype='dashed',color=
'blue',method = 'gam',se=F )+

geom_smooth(aes(x=mydata$Year,y=predict_2008[ ,'lwr']),linetype='dashed',color=
'red',method = 'gam',se=F)+

geom_smooth(aes(x=mydata$Year,y=predict_2008[ ,'upr']),linetype='dashed',color=
'red',method = 'gam',se=F)+
 geom_point(aes(x=accident$Year,accident$accdient_Rate),color="blue",size=3)+
  geom_smooth(aes(x = trainingData$Year, y = predict(lmMod, newdata =
trainingData),linetype =' Linear Moedl'),
            colour = 'blue',method = 'gam')+
  scale_x_discrete(limits =c(2005:2017))+ scale_y_discrete(limits
=c(1:15))+ggtitle('Note:Dashed Line predicted interval\n uper,fit,lower
value')+geom_text(aes(x = mydata$Year, y =
predict_2008[ ,'fit'],label=paste0(round(predict_2008[ ,'fit'],1),"%")),vjust=-
0.8,size=5,color="DarkGreen")
```

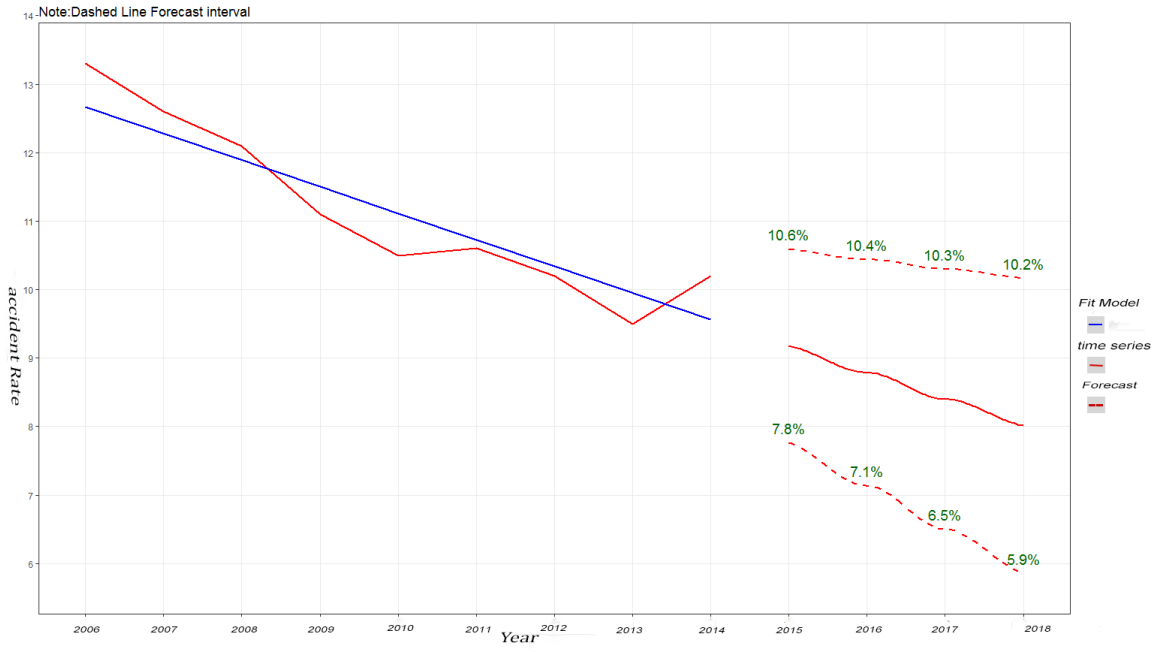


FIGURE5.6:FORECASTING ACCIDENT RATE

5.3.6 -Prediction interval confidence interval: to predict new value not invloved in the model.

Confidence Interval on New Observations assume that a new value of the yield is observed after the regression model is fit to the data. This new observation is independent of the observations used to obtain the regression model. If is the level of the temperature at which the new observation was taken.

For a confidence interval needs to be obtained on, then this interval should include both the error from the fitted model and the error associated with future observations. This is because represents the estimate for a value of that was not used to obtain the regression model. The confidence interval on is referred to as the prediction interval. A 100 () percent prediction interval on a new observation is obtained as follows:

5.4-Second: we predict accident Rate for each Month in The Year.

1-Time series decomposition:

Works by splitting a time series into three components: seasonality, trends and random fluctuation.

- Seasonal: Patterns that repeat with a fixed period of time. For example, a website might receive more visits during weekends; this would produce data with a seasonality of 7 days.

- Trend: The underlying trend of the metrics. A website increasing in popularity should show a general trend that goes up.

- Random: Also call “noise”, “irregular” or “remainder,” this is the residuals of the original time series after the seasonal and trend series are removed.

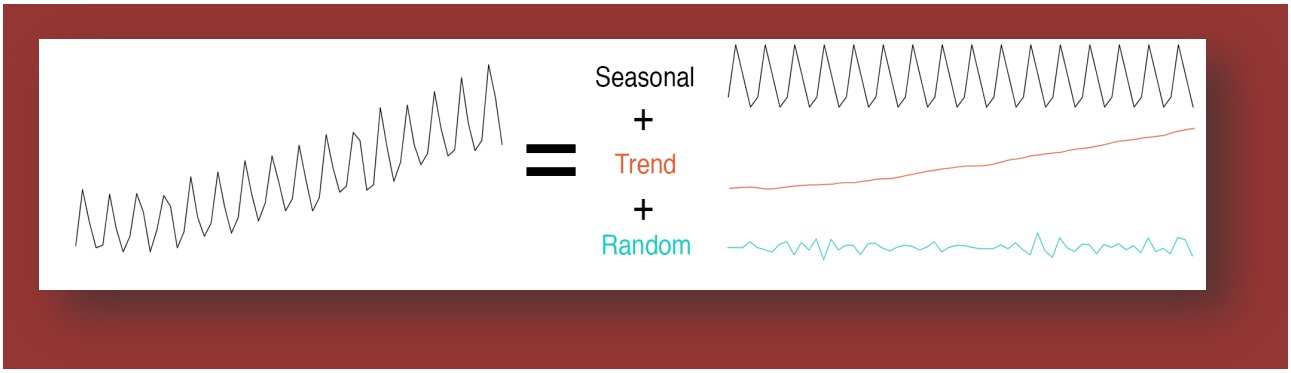
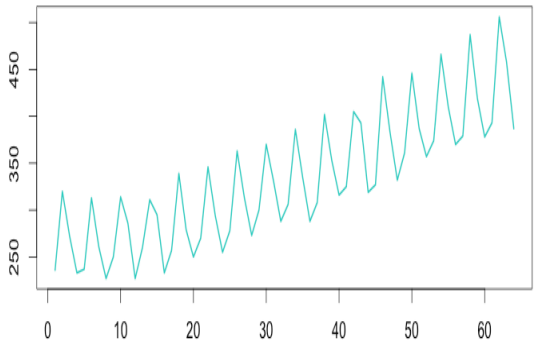


FIGURE5.7: TIME SERIES DECOMPISITION

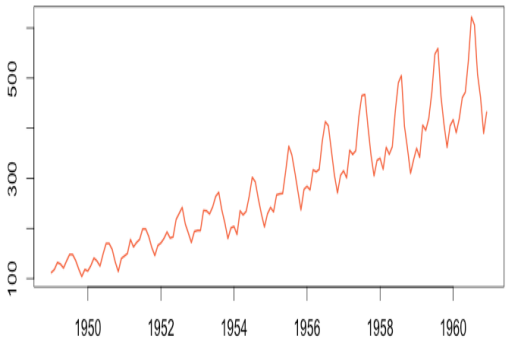
1.2 Additive or Multiplicative Decomposition?

To achieve successful decomposition, it is important to choose between the additive and multiplicative models, which requires analyzing the series. For example, does the magnitude of the seasonality increase when the time series increases?



Additive Model

FIGURE5.9: ADDITIVE MODEL



multiplicative Model.

FIGURE5.8: MULTIPLICATIVE MODEL

The seasonal variation looks constant;

It doesn't change when the time series value increases.

We should use the **additive model**

As the time series increases in magnitude, the seasonal variation increases as well. Here we should use the **multiplicative model**.

Additive:

Time series = Seasonal + Trend + Random

Multiplicative:

Time series = Seasonal *Trend *Random

2-Seasonal trend model with dummy variables.

Fit a Linear Model with Time Series components of Trend and Seasonality.

Seasonality in the regression world:

Now we need to model seasonality

In the linear regression world this involves dummy variables.

Can take many forms

Quarterly only? Monthly only? Easily combine with trends.

❖ Seasonal trend equation:

Linear trend + Monthly seasonal pattern

Multiple regressions with time trend (month = 1, 2, 3...) and monthly dummy variables (11 indicators, Dec omitted)

$$Yt = b0 + \beta trend + b1jan + b\ 2feb + b\ 3mar + b\ 4apri \ldots\ldots + b12Desc + ut.$$

Where Trend = 1 in year 1

= 2 in year 2

= T in year T

❖ **Goal: Forecast seasonal accident Rate per Year.**

2- Preparing the Data.

❖ Select Date.

```
head(accidents$Date)
```

```
## [1] 05/01/2005 13/01/2005 15/01/2005 15/01/2005 16/01/2005 25/01/2005
## 3286 Levels: 01/01/2005 01/01/2006 01/01/2007 01/01/2009 ... 31/12/2014
```

❖ Choose only month from Date.

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
accidents<-mutate(accidents,month=month(Date))
head(accidents$month,n=10)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
```

Group accidents by year.

```
k<-group_by(accidents,Year,month)
```

❖ Summaries by years to find total number of accidents in each month of year.

```
k<- dplyr::summarize(k,accidents = n())
```

```
head(k,n=20)
```

```
## # A tibble: 20 x 3
## # Groups:   Year [2]
##     Year month accidents
##    <int> <dbl>     <int>
##  1  2005  1.00      9500
##  2  2005  2.00      8506
##  3  2005  3.00      8793
##  4  2005  4.00      9277
```

```
##  5  2005  5.00     10172
##  6  2005  6.00      9895
##  7  2005  7.00      9826
##  8  2005  8.00      9545
##  9  2005  9.00      9917
## 10  2005 10.0      10300
## 11  2005 11.0      10913
## 12  2005 12.0       9743
## 13  2006  1.00      8680
## 14  2006  2.00      8143
## 15  2006  3.00      8808
## 16  2006  4.00      8095
## 17  2006  5.00      9497
## 18  2006  6.00      9304
## 19  2006  7.00      9743
## 20  2006  8.00      8767
```

❖ Compute percent % for accidents in each month of year.

```
accident<-select(accident,c(Year:accdient_Rate))

head(accident)
```

```
##   Year month accidents accdient_Rate
## 1 2005     1      9500           8.3
## 2 2005     2      8506           7.9
## 3 2005     3      8793           8.3
## 4 2005     4      9277           9.9
## 5 2005     5     10172          11.4
## 6 2005     6      9895          10.4
```

3- Creating time series of Monthly accident Rate.

3.1 For each Year.

```
myts <- ts(accident$accdient_Rate, frequency=12, start = c(2005,1), end =
c(2014,12))


myts
```

```
##       Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
## 2005  8.3  7.9  8.3  9.9 11.4 10.4 10.8 11.6 10.8  9.0  9.5  9.1
## 2006  9.0  8.6  9.4  8.8 11.6 10.8  8.3  7.9  9.1 10.8 11.4 10.4
## 2007  9.8  9.5  9.8  7.5  8.7  8.3  9.0  9.5  9.4  9.8 11.6  9.8
## 2008  6.8  5.6  7.4  8.1  8.6  9.4  9.8  9.5  9.8  7.5  8.7  7.4
## 2009  6.3  7.6  8.5  7.8  9.5  9.8  6.8  7.1  8.3  9.0  9.5  5.7
## 2010  7.8  8.4  8.8  6.8  7.1  7.4  8.1  7.6  8.5  9.8  9.5  8.8
## 2011  6.0  6.3  7.4  7.2  8.6  7.5  8.8  8.4  8.8  6.8  7.1  6.6
## 2012  6.3  6.7  6.6  6.9  8.4  7.8  6.8  6.3  6.6  8.1  8.6  7.5
## 2013  8.8  8.4  7.8  6.0  7.1  7.4  8.1  7.6  7.5  9.8  9.5  7.8
## 2014  8.3  7.9  8.3  9.9 11.4 10.4 10.8 11.6 10.8  9.0  9.5  9.1
```

❖ Plot Time series Data for Monthly accident Rate.

```
 plot(myts,col='red')
axis(1, at=2005:2015)

box()
```

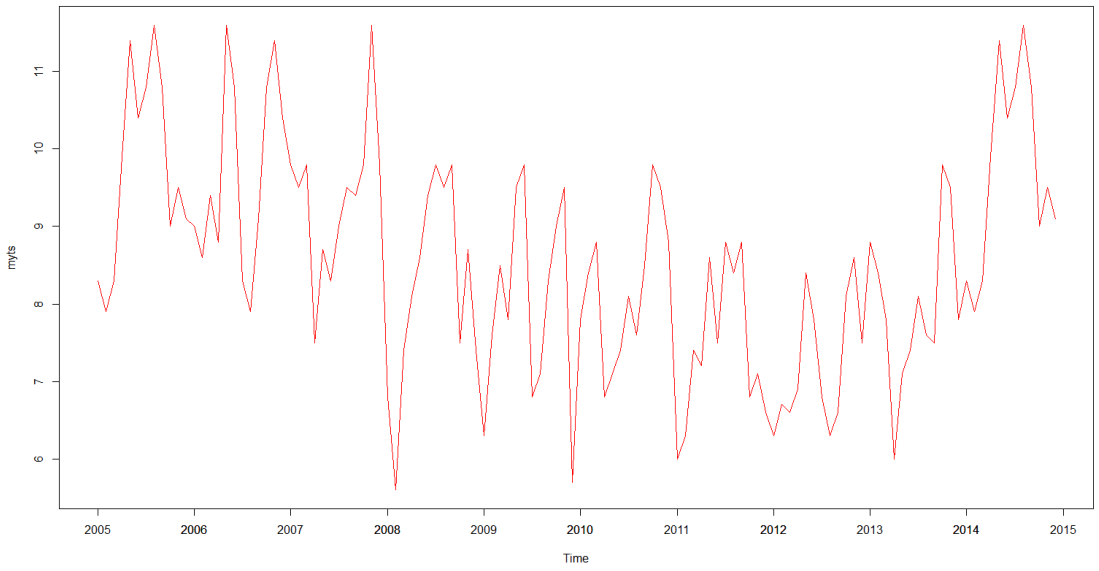


FIGURE5.10:TIME SERIES DATA FOR MONTHLY ACCIDENT RATE

❖ Convert Time series to data frame.

```
my_df_ts <- data.frame(accident_Rate= myts, as.numeric(time(myts)))
 names(my_df_ts) <- c("accident_Rate", "time")
```

❖ Decompose different periods of time series trend and seasonal.

Use Multiplicative if data set has INCREASING or DECREASING Seasonality variation

```
myds_month <- decompose(myts, "multiplicative")
plot(myds_month,col='red')
axis(1, at=2005:2015)

box()
```

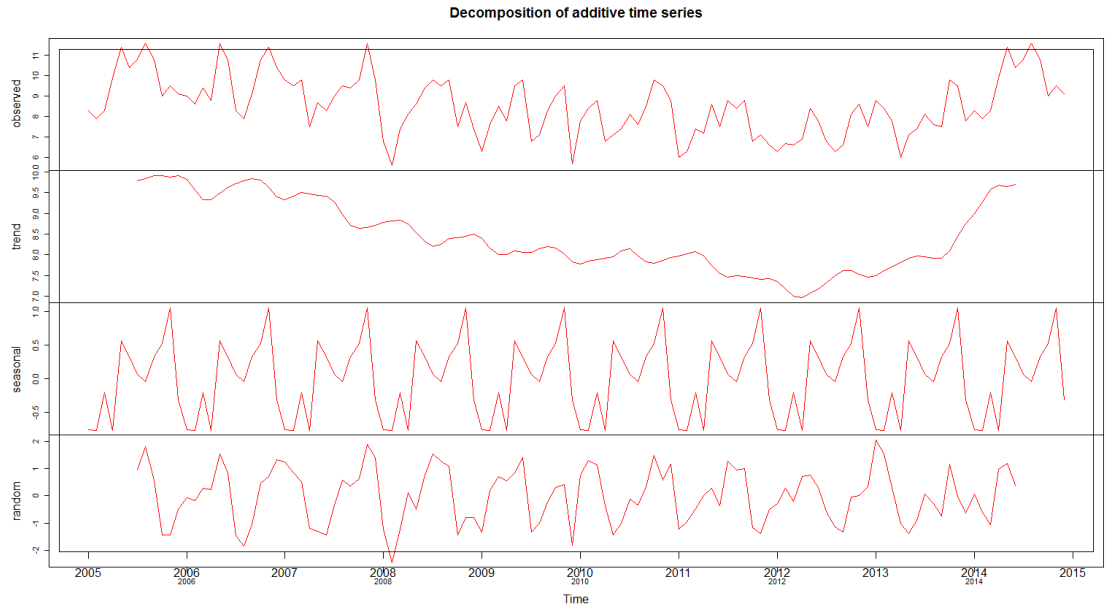


FIGURE5.11: DECOMPOSE DIFFERENT PERIODS OF TIME SERIES TREND AND SEASONAL.

4- Build Seasonal trend Model.

```
 mymodel <- tslm(accident_Rate~season+trend,my_df_ts)

summary(mymodel)

##
## Call:
## tslm(formula = accident_Rate ~ season + trend, data = my_df_ts)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -2.5951 -0.9652 -0.0749  0.8046  3.5655
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.427963   0.457013  18.441  < 2e-16 ***
## season2     -0.037492   0.587037  -0.064 0.949196
## season3      0.515017   0.587068   0.877 0.382305
## season4      0.187525   0.587119   0.319 0.750047
## season5      1.550034   0.587191   2.640 0.009537 **
## season6      1.242542   0.587284   2.116 0.036687 *
## season7      1.065051   0.587397   1.813 0.072609 .
## season8      1.057559   0.587531   1.800 0.074677 .
## season9      1.320067   0.587685   2.246 0.026744 *
## season10     1.332576   0.587860   2.267 0.025411 *
## season11     1.875084   0.588055   3.189 0.001875 **
## season12     0.617593   0.588271   1.050 0.296155
## trend       -0.012508   0.003477  -3.598 0.000487 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.313 on 107 degrees of freedom
## Multiple R-squared:  0.2505, Adjusted R-squared:  0.1664
## F-statistic:  2.98 on 12 and 107 DF,  p-value: 0.001269
```

5-forecast Monthly accident Rate for 2016, 2017, 2018.

```
 my_fc <- forecast(mymodel,h=36,level=95)

my_fc
```

```
##          Point Forecast    Lo 95     Hi 95
## Jan 2015       6.914444 4.147654  9.681235
## Feb 2015       6.864444 4.097654  9.631235
## Mar 2015       7.404444 4.637654 10.171235
## Apr 2015       7.064444 4.297654  9.831235
## May 2015       8.414444 5.647654 11.181235
## Jun 2015       8.094444 5.327654 10.861235
## Jul 2015       7.904444 5.137654 10.671235
## Aug 2015       7.884444 5.117654 10.651235
## Sep 2015       8.134444 5.367654 10.901235
## Oct 2015       8.134444 5.367654 10.901235
## Nov 2015       8.664444 5.897654 11.431235
## Dec 2015       7.394444 4.627654 10.161235
## Jan 2016       6.764343 3.982761  9.545926
## Feb 2016       6.714343 3.932761  9.495926
## Mar 2016       7.254343 4.472761 10.035926
## Apr 2016       6.914343 4.132761  9.695926
```

```
## May 2016       8.264343 5.482761 11.045926
## Jun 2016       7.944343 5.162761 10.725926
## Jul 2016       7.754343 4.972761 10.535926
## Aug 2016       7.734343 4.952761 10.515926
## Sep 2016       7.984343 5.202761 10.765926
## Oct 2016       7.984343 5.202761 10.765926
## Nov 2016       8.514343 5.732761 11.295926
## Dec 2016       7.244343 4.462761 10.025926
## Jan 2017       6.614242 3.815501  9.412984
## Feb 2017       6.564242 3.765501  9.362984
## Mar 2017       7.104242 4.305501  9.902984
## Apr 2017       6.764242 3.965501  9.562984
## May 2017       8.114242 5.315501 10.912984
## Jun 2017       7.794242 4.995501 10.592984
## Jul 2017       7.604242 4.805501 10.402984
## Aug 2017       7.584242 4.785501 10.382984
## Sep 2017       7.834242 5.035501 10.632984
## Oct 2017       7.834242 5.035501 10.632984
## Nov 2017       8.364242 5.565501 11.162984
## Dec 2017       7.094242 4.295501  9.892984
```

❖ Plot forecast Data for 2015 to 2018.

```
 plot(my_fc,col='red')
axis(1, at=2005:2015)

box()
```

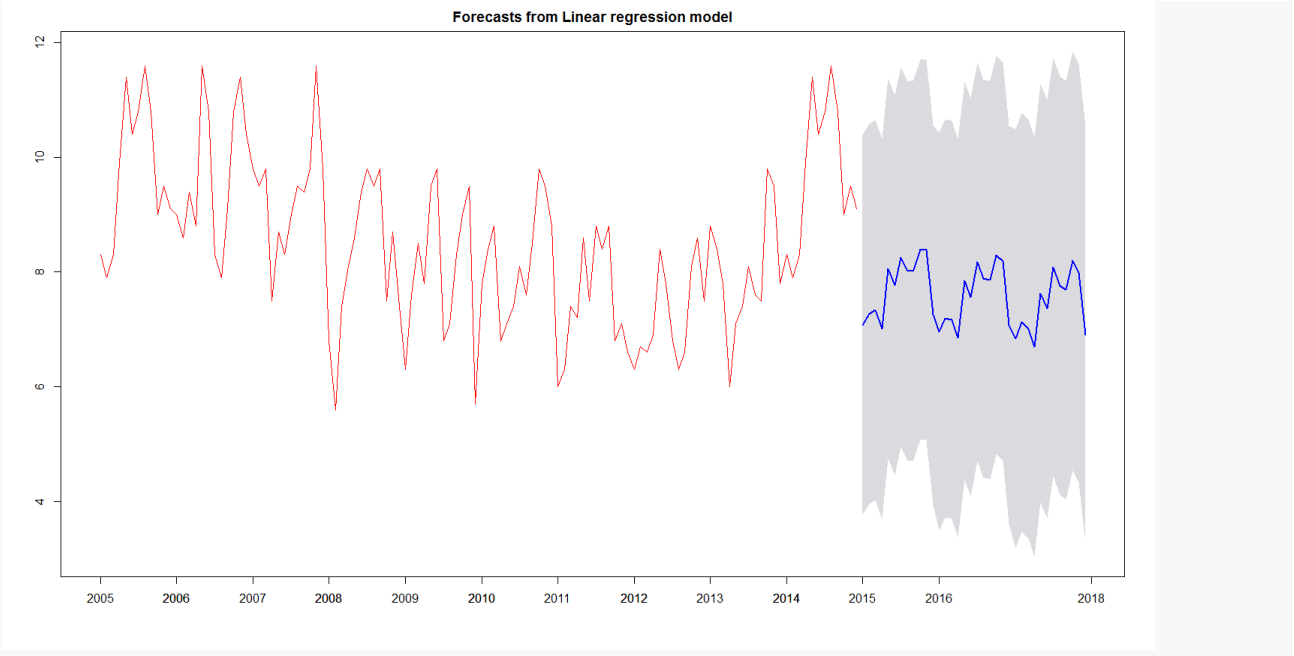


FIGURE5.12: FORCAST FROM LINEAR REGRESSION MPDEL

6

chapter six

Web Shiny

- Introduction about shiny
- Building shiny Aplication
- Aplication purpose
- Why Make Dashbord with shiny?
- Functionality
- How to use App

6.1 Introduction about Shiny

Shiny is an R package that makes it easy to build interactive web apps straight from R, you can host standalone apps on web pages or embed them in R Markdown documents or build dashboards. You can also extend

Our Shiny apps with CSS themes ,html widgets ,and JavaScript actions

Shiny combines the computational power of R with the interactivity of the modern web .Shiny apps are easy to write .No web development skills are required

With the advent of data science and the increased need to analyse and interpret vast amounts of data, the R language has become ever more popular. However, there's increasingly a need for a smooth interaction between statistical computing platforms and the web, given both 1) the need for a more interactive user interface in analyzing data, and 2) the increased role of the cloud in running such applications.

Statisticians and web developers have thus seemed an unlikely mix till now, but make no mistake that the interactions between these two groups will continue to increase as the need for web-based platforms becomes ever more popular in the world of data science. In this regard, the interaction of the R and Shiny platforms is quickly becoming a cornerstone of interaction between the world of data and the web. (shiny, 2017)

6.2 Building Shiny APP:

The structure of a web-application in generally takes the following form:

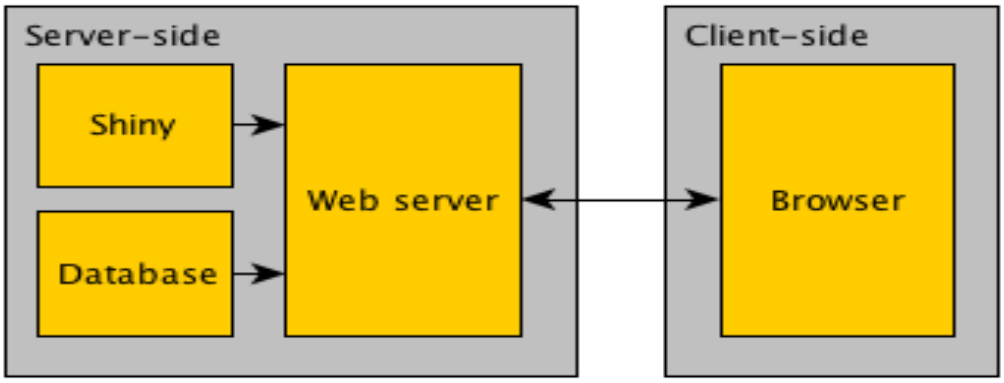


FIGURE6.1 : THE STRUCTURE OF WEB APPLICATION

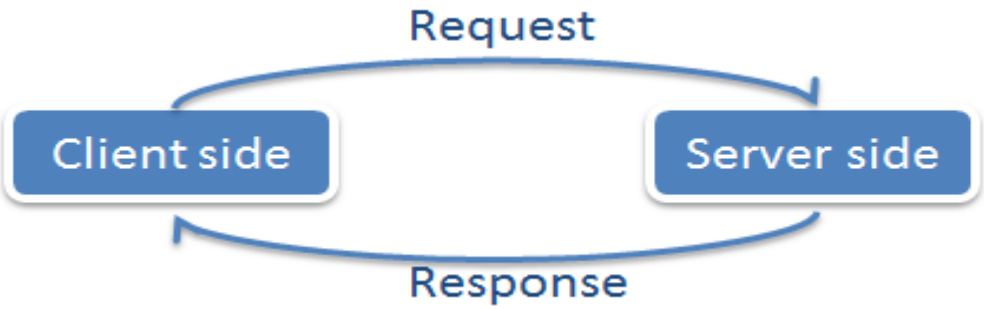
A Shiny application is simply a directory containing:

- User-interface definition
- Server script.
- Other resources required to support the application.

6.3 Shiny App Structure:

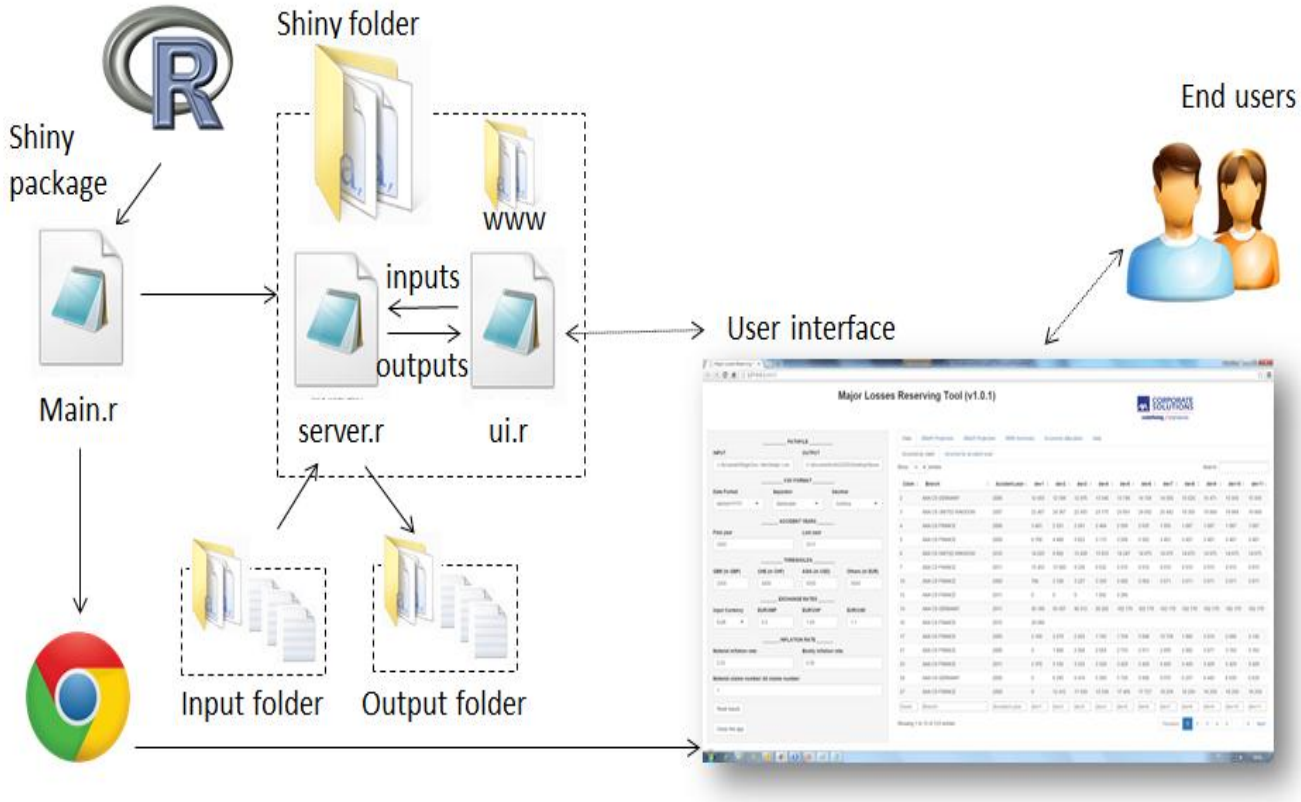
The logic of Shiny relies on the two notions that are very close to the web ones: client side and server side. Client side is the component in interaction with the end-users. Server side is the component through what your R program receives requests from users, runs R code and then send the response to the client side. (Data Visualization Shiny Application, 2017)

figure6.2: UI&server



❖ Shiny environment

FIGURE6.3: SHINY ENVIRONMENT



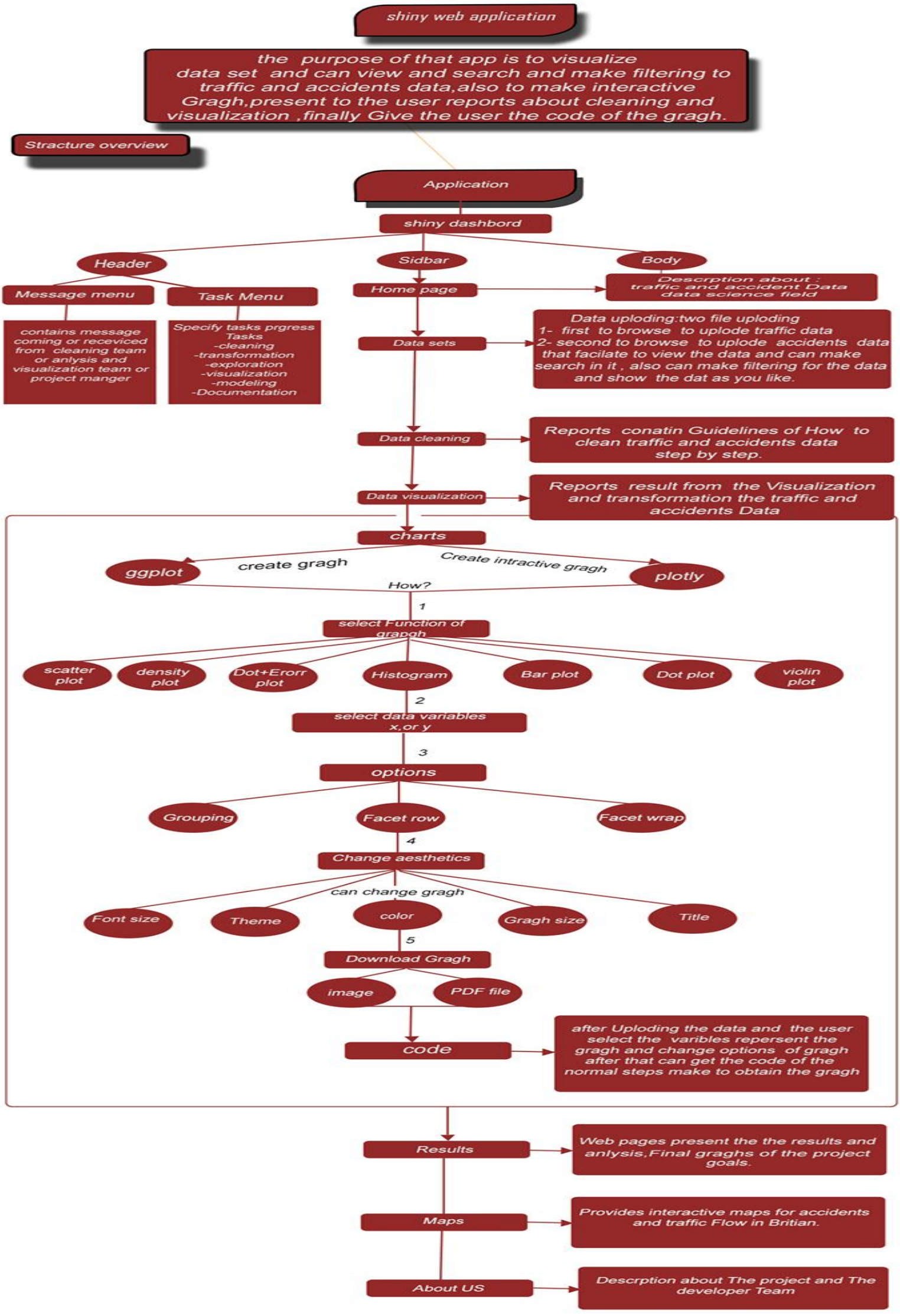
Based on this principle, Shiny requires 2 R scripts: one for the User Interface, named ui., and one for the Server, named server.r. What you want to display on the the interface needs to be declared in the ui.r script; how this thing is done needs to be defined in server.r. (Data Visualization Shiny Application, 2017)

6.4 Application purpose

❖ Uploading data and can view it, also can filter data and make search in it easily.

- Facilitate to make interactive graphs for the data.
- Present user or cleaning team reports and Guidelines about how the accidents and traffic data were cleaned step by step.
- Present user or visualization team reports and Guidelines about how the accidents and traffic data were visualized and results.
- Can make interactive graphs easily to the data and can download the graphs and interact with it.
- Can update the graphs and change themes, colour, text titles, desertion of graphs.
- It easily can get the code of graphs.
- Present the results of analysis of traffic and accident data.
- Interactive maps for the accidents data and traffic flow in Britain.

6.5 Application Structure


shiny web application
the purpose of that app is to visualize data set and can view and search and make filtering to traffic and accidents data,also to make interactive Gragh,present to the user reports about cleaning and visualization ,finally Give the user the code of the gragh.
Stracture overview
Application
shiny dashbord
Header
Sidbar
Body
Message menu
Task Menu
Home page
Descrption about : traffic and accident Data data science field
contains message coming or receviced from cleaning team or anlysis and visualization team or project manger
Specify tasks prgress Tasks -cleaning -transformation -exploration -visualization -modeling -Documentation
Data sets
Data uploding:two file uploding 1- first to browse to uplode traffic data 2- second to browse to uplode accidents data that facilate to view the data and can make search in it , also can make filtering for the data and show the dat as you like.
Data cleaning
Reports conatin Guidelines of How to clean traffic and accidents data step by step.
Data visualization
Reports result from the Visualization and transformation the traffic and accidents Data
charts
ggplot
create gragh
Create intractive gragh
plotly
How?
1
select Function of grapgh
scatter plot
density plot
Dot+Erorr plot
Histogram
Bar plot
Dot plot
violin plot
2
select data variables x,or y
3
options
Grouping
Facet row
Facet wrap
4
Change aesthetics
can change gragh
Font size
Theme
color
Gragh size
Title
5
Download Gragh
image
PDF file
code
after Uploding the data and the user select the varibles repersent the gragh and change options of gragh after that can get the code of the normal steps make to obtain the gragh
Results
Web pages present the the results and anlysis,Final graghs of the project goals.
Maps
Provides interactive maps for accidents and traffic Flow in Britian.
About US
Descrption about The project and The developer Team

6.6 Why Make Dashboards with Shiny?

There's three very good reasons…R is a programming language…with an ecosystem built for data analysis, Modelling, machine learning, and data science in general….Shiny allows you to incredibly easily convert these script files into interactive applications By integrating your analytical scripts directly into the dashboard workflow you're ensuring your dashboards are always up to date, and that your data experts don't depend on different tools for analysis and communication of results.

Multi-page responsive dashboard can easily be built in Shiny and natively supports the Bootstrap framework.

Run and displaying:

The user interface can be displayed on

- ❖ The default browser of RStudio if you run Shiny through it.
- ❖ On a web browser.

This user interface works exactly like a web page. There are buttons, input text, etc. for you to make your decision and there will be output (graphs, images, tables, etc.) displayed on the main panel, or separately on different tabs/panels.

6.7 Functionality:

Shiny dashboard consists of three main parts:

- Header
- Sidebar
- Body

Header description:

Contains messages coming or received from cleaning team, analysis and visualization team or project manager.

A message menu looks like this:

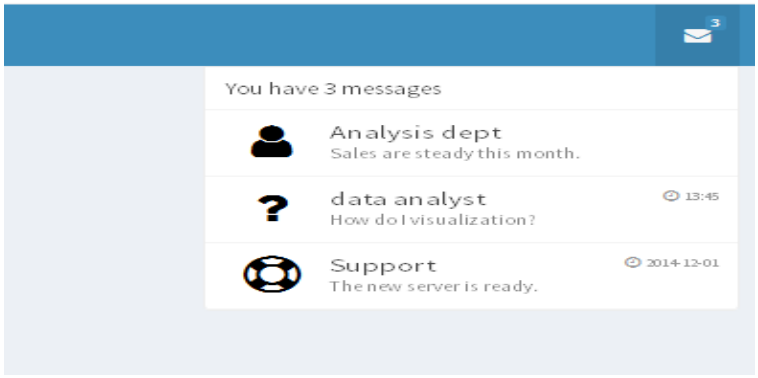


FIGURE6.4: MESSAGE MENU

Task Menu

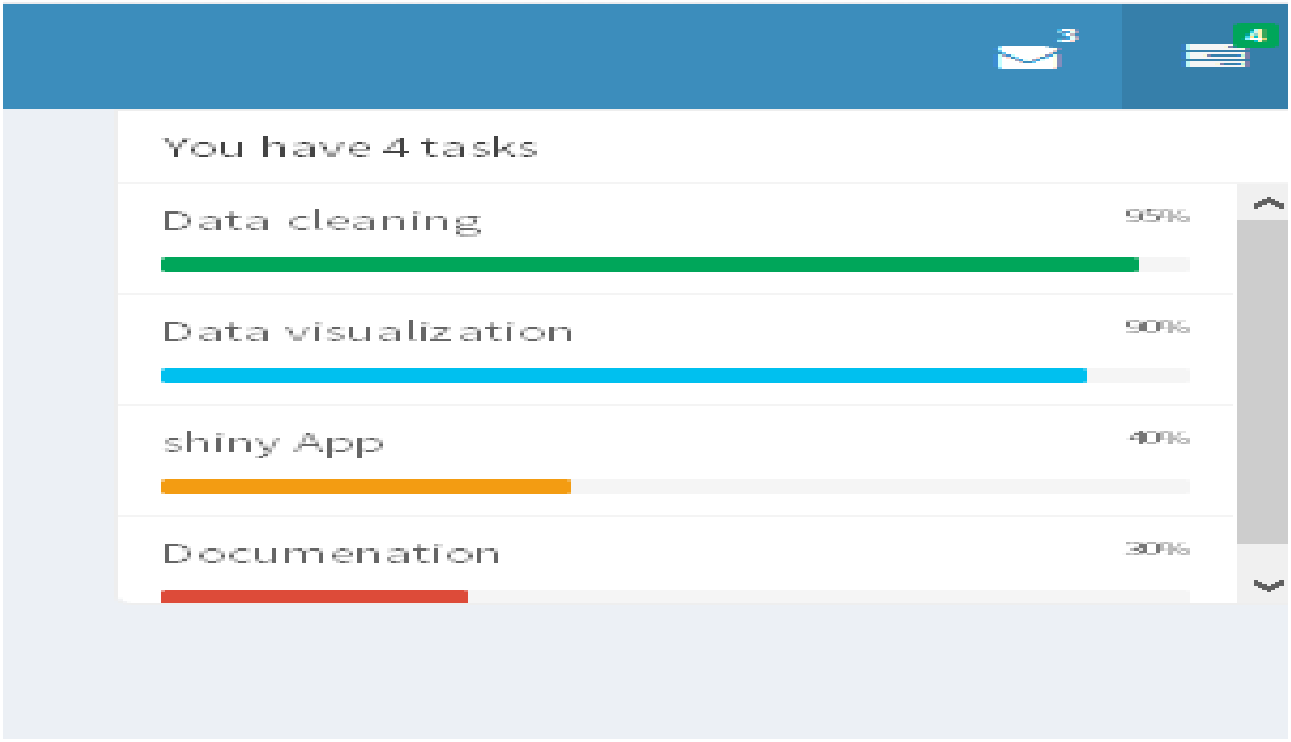
Task menu that specify tasks progress.

Description tasks are:

- Cleaning
- Transformation
- Exploration

- Visualization
- Modeling
- Documentation

Task menu Looks like this:

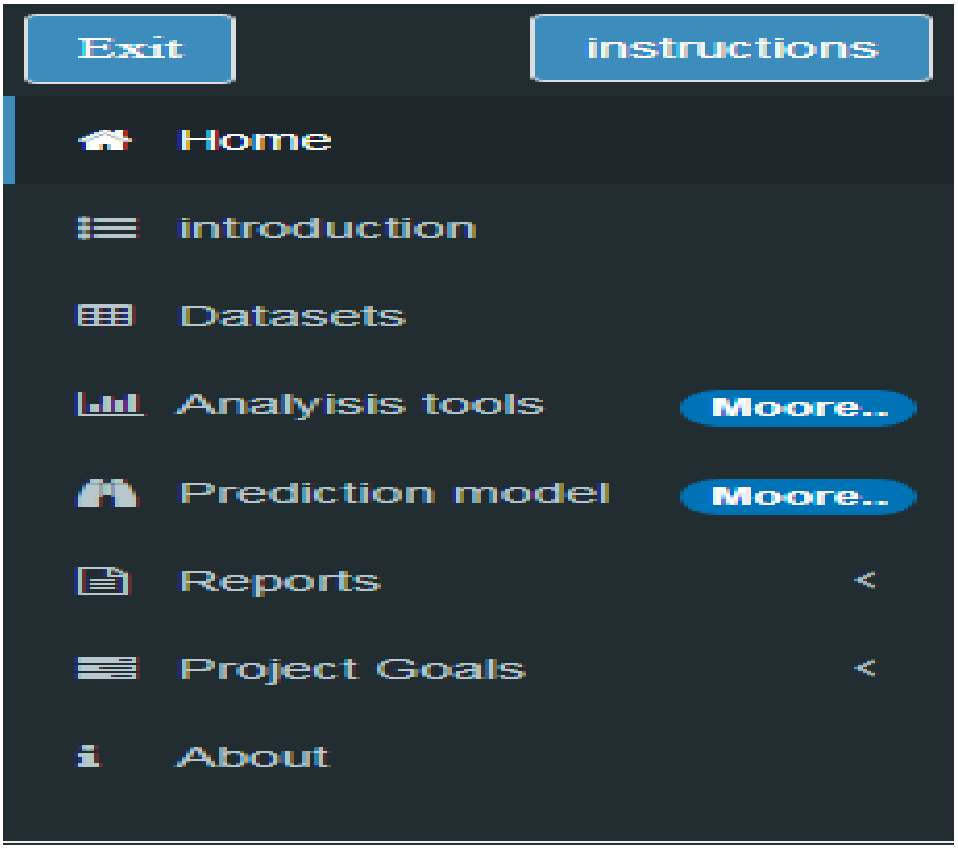


Sidebar

A sidebar is typically used for quick navigation. It can contain menu items that behave like tabs in a tab Panel, as well as Shiny inputs, like sliders and text input.

Sidebar consist of:

- Sidebar Menu
- Bookmarking and restoring selected tabs.
- Dynamic content.
- Inputs in side bar.
- Home page



- Uploading data sets

Sidebar Menu

It is Links in the sidebar can be used like tab Panels from Shiny.

That is when you click on a link, it will display different content in the body of the dashboard.

Consist of:

- Menu items…Included in sidebar:

  Menu items are put in sidebar Menu ()

- Tab items….included in body.

FIGURE6.5: SIDEBAR MENU

6.7 How to use application

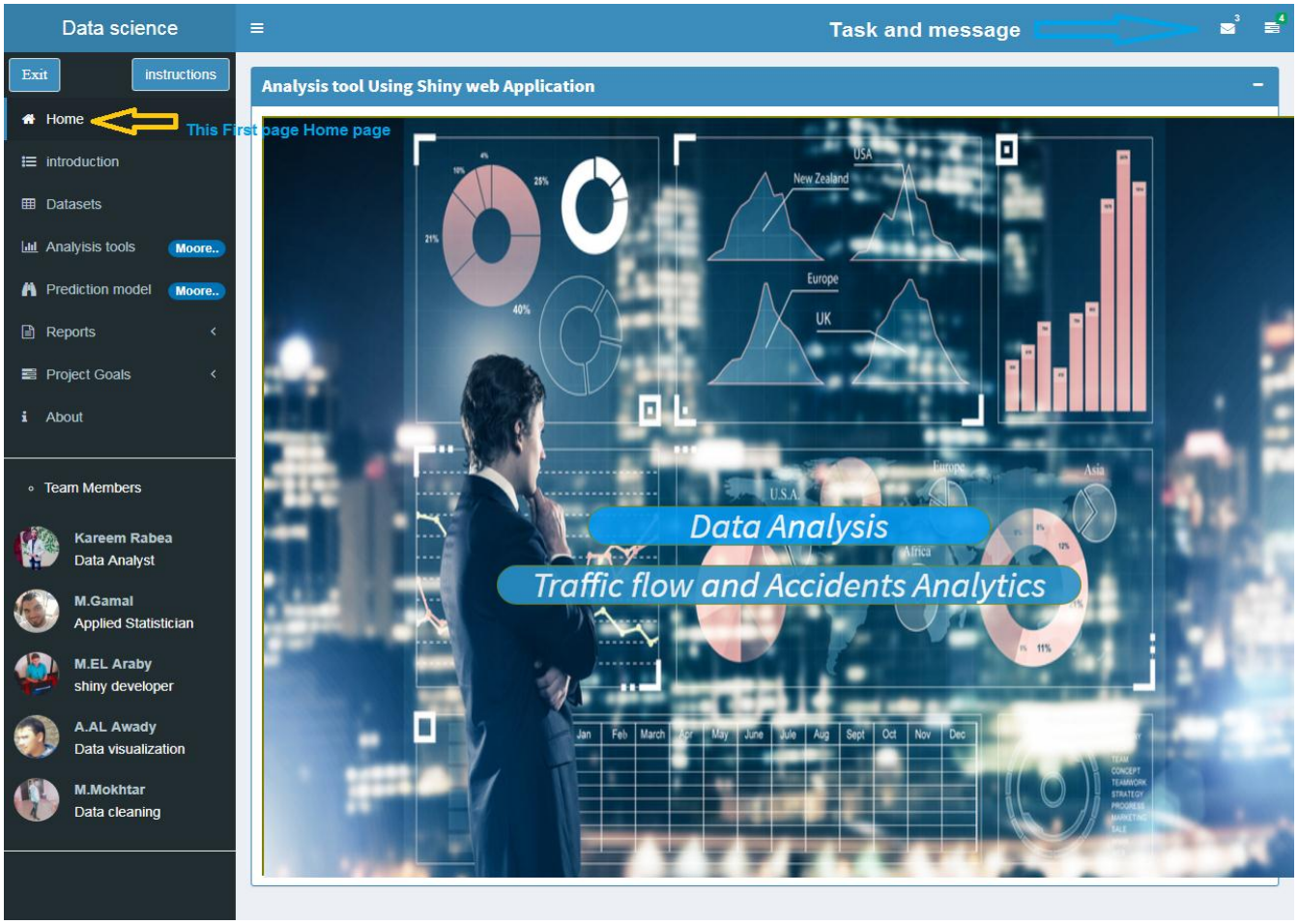
6.7.1 User's guide

For our customer we will make a simple user's guide for them to show how users can use the app, upload datasets, make graphics, see our cleaning and visualization documents, make interactive maps and read about us "owners of the project".

First when the user runs the app, it opens in browser or in new window of Rstudio.

Home page is the first page appears that contain information about data science field, with the header which contain messages and task menu.

6.7.2 This is the first page appears:



If you want to make graphs and visualization of data, you must upload data sets first from datasets element from sidebar.

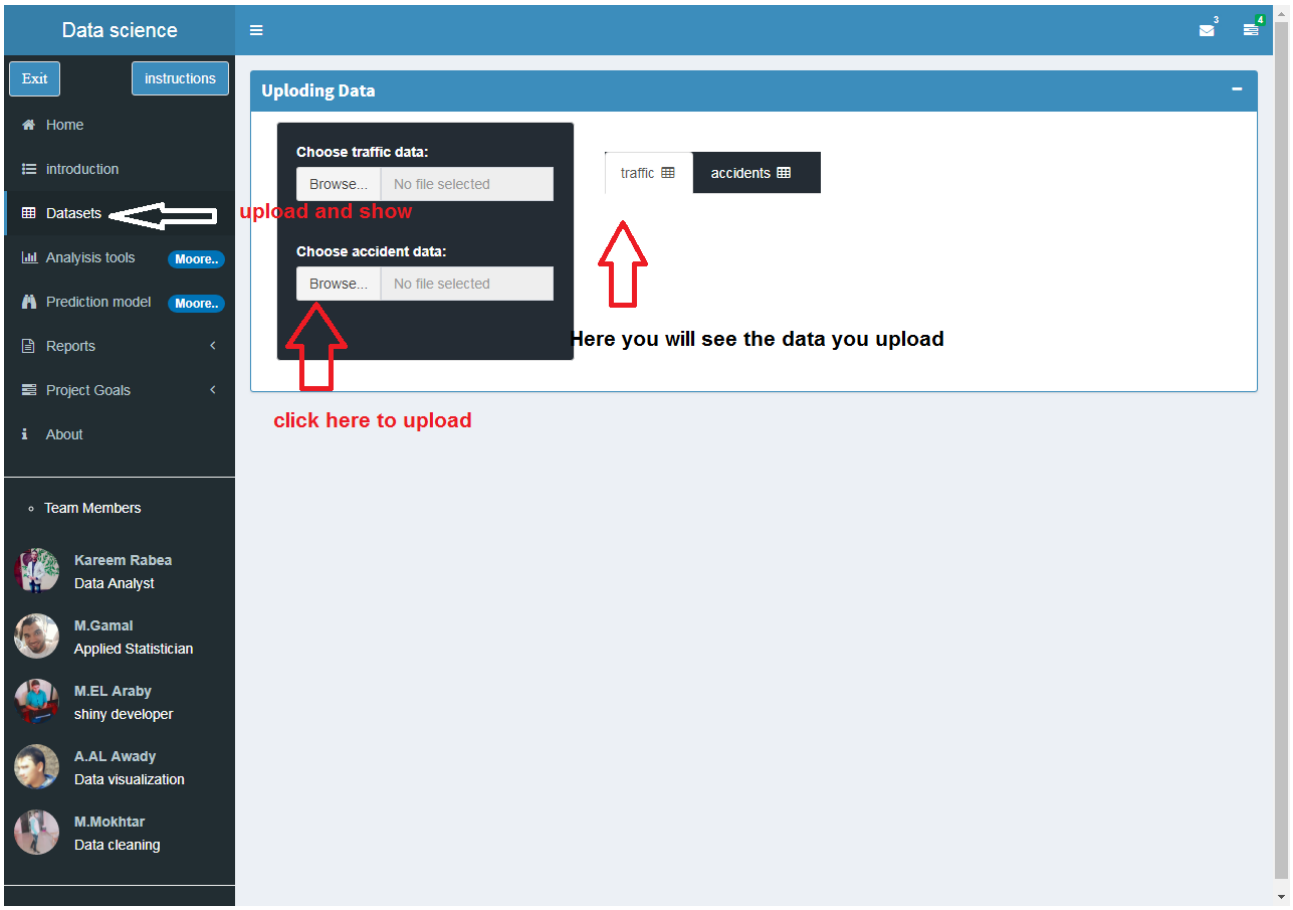
Then upload datasets from browse data.

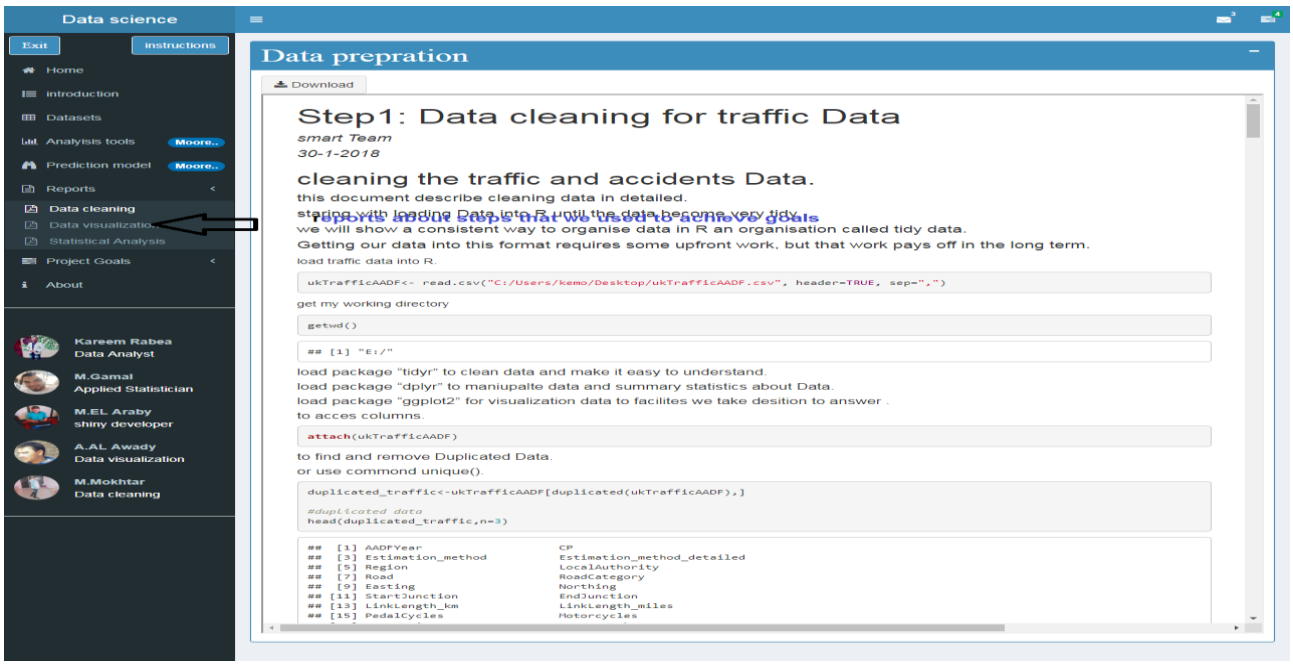
As showing in following page:

After selecting datasets, datasets appear with variables of each of dataset.

You can make search or choose number of rows to show.

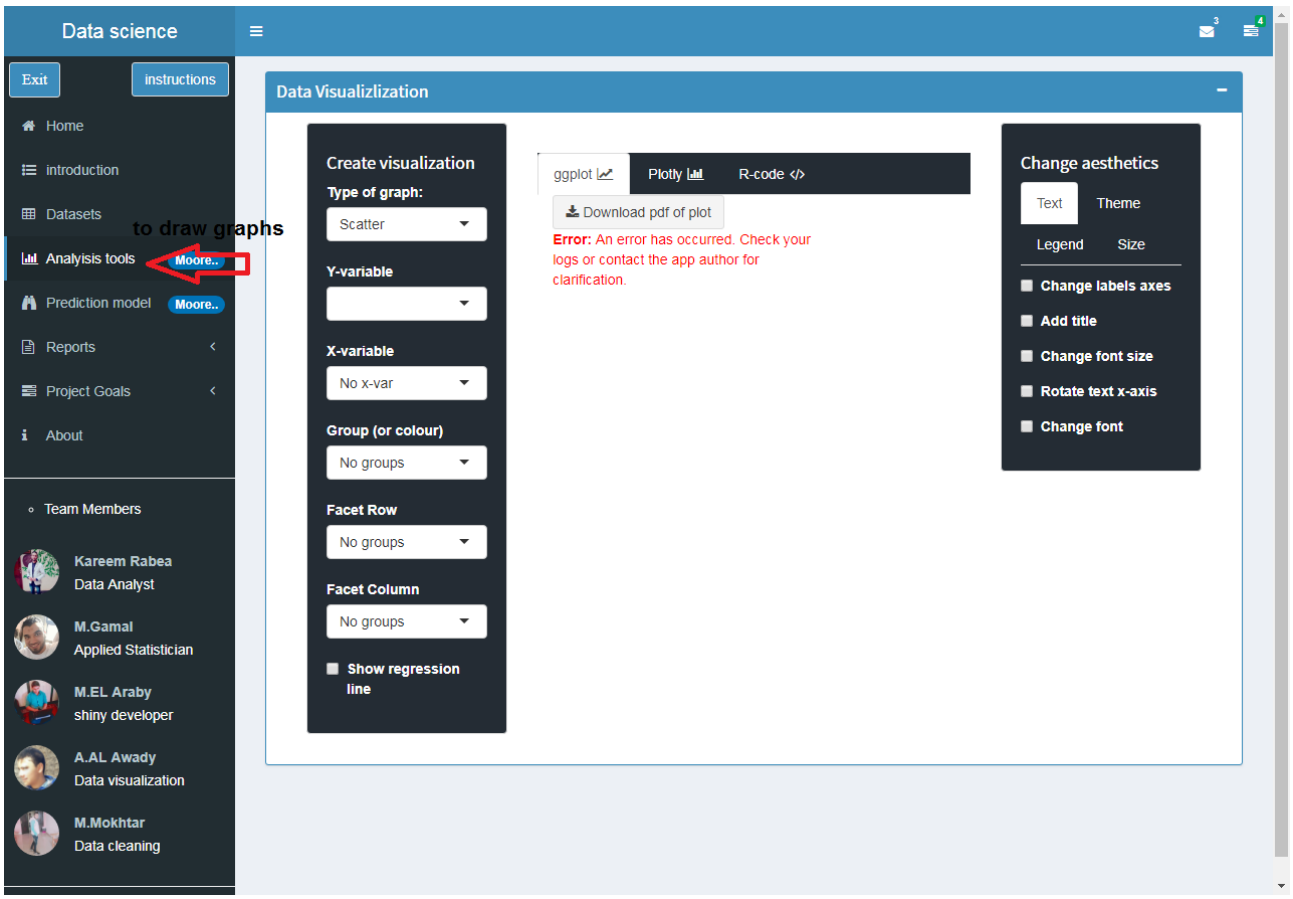
As following:





In reports element in side bar you can see data cleaning documentation, data visualization documentation and statical analysis.

In analysis tools element you can make graphs with analysis to datasets by choosing type of graph, type of function, names of variables and other options as facets and group or color,you can add title to graph, change labels axis, change font size, and rotate text of X axis,You can also find code of your graph in the code section.



In project goals you will find the goals we work hard to achieve in this project as it will be shown in the next chapter.

Data science

Exit | instructions

- Home
- introduction
- Datasets
- Analysis tools Moore..
- Prediction model Moore..
- Reports
- Project Goals
  - First Problem
  - Second Problem
  - Third Problem:
  - Fourth Problem:
  - Fifth Problem:
- About

Goals we achieve

Kareem Rabea
Data Analyst

M.Gamal
Applied Statistician

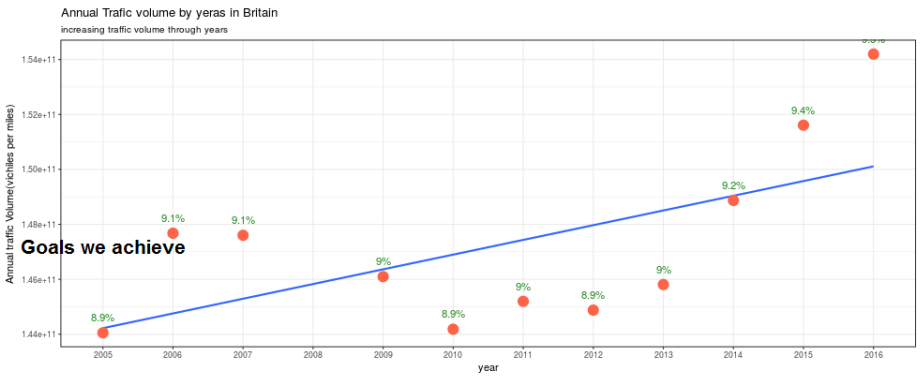
M.EL Araby
shiny developer

A.AL Awady
Data visualization

M.Mokhtar
Data cleaning

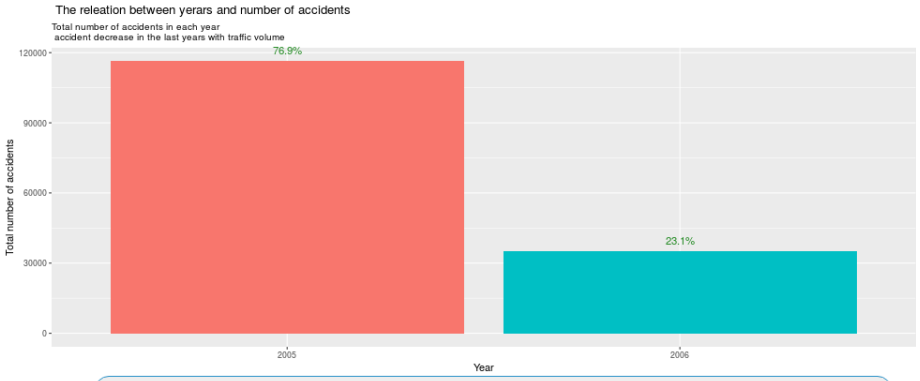
How has changing traffic flow impacted accidents?

1.1-Annual Trafic volume by yeras in Britain



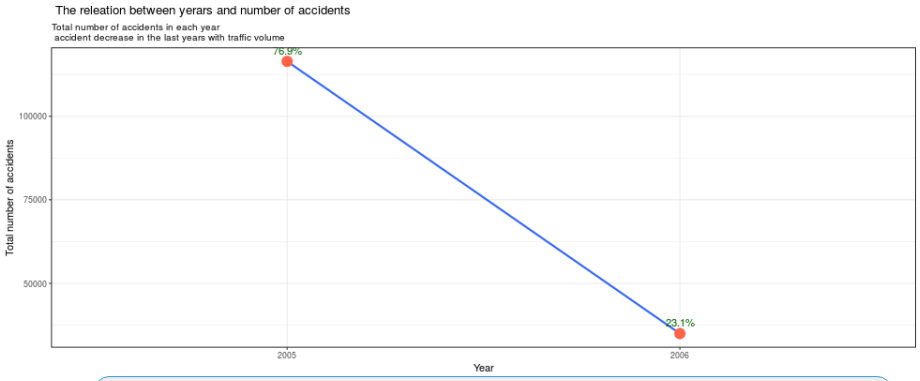
interpreation: The gragh show that the traffic flow increasing with the Years and show that 2016 the large year fo traffic rate approximately 9.1% from the whole traffic low in UK.

1.2-The Relation between Year and accidents Rate



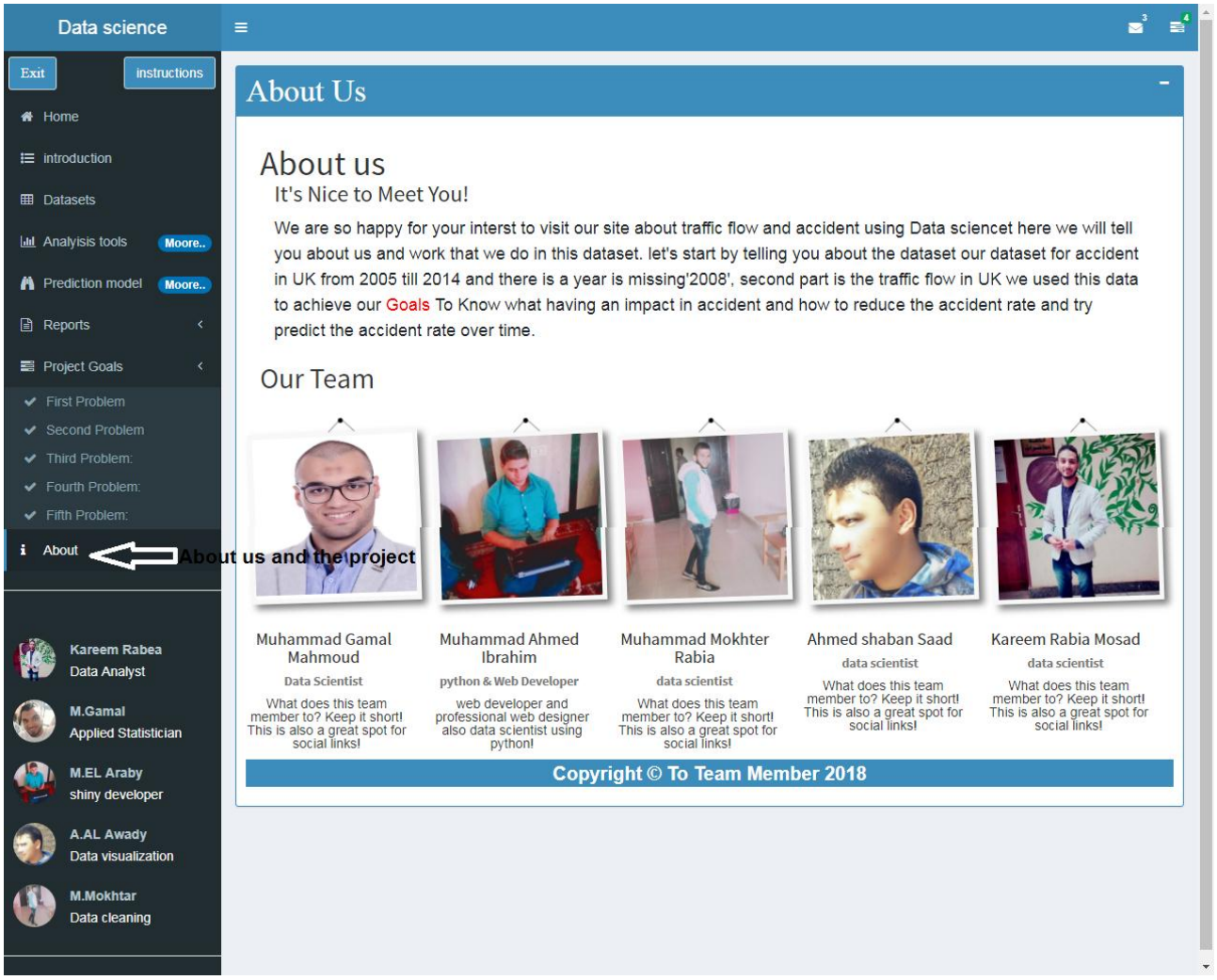
interpreation: The gragh show that taccidents Rate Decrease with the Years and show that 2014 has accidents rate is smaller than the whole accidents rate throgh Years in UK.

1.3-The Relation between Year and accidents Rate



interpreation: The gragh show that accidents Rate Decrease with the Years and show that 2014 has accidents rate is smaller than the whole accidents rate throgh Years in UK.

In about us element you will know more about the team of the project and explansion about the project.



7

chapter seven

Implementation

- pre-processing cleaning and manuplation process
- Project Goal
- **Shiny implementation**

First step in my project:

7.1 Clean data and manipulation process :(pre-processing data).

```
library(dplyr)
library(tidyr)
library(ggplot2)
```

7.1.1- read data (csv files)

"Acc" is an object include the accidents data which we work on, divided into specified and different period,For prevent character columns to convert to factor we use argument "stringsAsFactors = FALSE".

```
acc_2005_2007 <- read.csv("data/accidents_2005_to_2007.csv",stringsAsFactors = FALSE)
acc_2009_2011 <- read.csv("data/accidents_2009_to_2011.csv",stringsAsFactors = FALSE)
acc_2012_2014 <- read.csv("data/accidents_2012_to_2014.csv",stringsAsFactors = FALSE)
```

7.1.2 Read data as a tabular data

To can work on data useing dplyr functions, data must be in tabular form.

```
acc_05_07 <- tbl_df(acc_2005_2007)
acc_09_11 <- tbl_df(acc_2009_2011)
acc_12_14 <- tbl_df(acc_2012_2014)
```

We can do last two process (read data and convert to a data frame) in one comand as following: By this way we not need to remove original data to can transform data.

"TrafficAADF" object incluse the traffic flow records.

```
TrafficAADF <- tbl_df(read.csv("data/ukTrafficAADF.csv", stringsAsFactors = FALSE))
```

7.1.3 We will remove the original data

To save data and prevent confused while making operation, must be remove original data.

We can remove only specified object as following:

```
rm("acc_2012_2014")
```

Or remove all objects which i want to remove as a list in one command.

```
rm(list = c("acc_2005_2007","acc_2009_2011"))
```

7.1.4 We look at the head (first six rows) of each data object

A head of accident data from 2005 to 2007

```
head(acc_05_07)
```

```
## # A tibble: 6 x 33
##   Accident_Index Location_Easting_O… Location_Northing… Longitude Latitude
##   <chr>                        <int>              <int>     <dbl>    <dbl>
## 1 200501BS00001                525680             178240    -0.191     51.5
## 2 200501BS00002                524170             181650    -0.212     51.5
## 3 200501BS00003                524520             182240    -0.206     51.5
## 4 200501BS00004                526900             177530    -0.174     51.5
## 5 200501BS00005                528060             179040    -0.157     51.5
## 6 200501BS00006                524770             181160    -0.203     51.5
## # ... with 28 more variables: Police_Force <int>, Accident_Severity <int>,
## #   Number_of_Vehicles <int>, Number_of_Casualties <int>, Date <chr>,
## #   Day_of_Week <int>, Time <chr>, Local_Authority_.District. <int>,
## #   Local_Authority_.Highway. <chr>, X1st_Road_Class <int>,
## #   X1st_Road_Number <int>, Road_Type <chr>, Speed_limit <int>,
## #   Junction_Detail <lgl>, Junction_Control <chr>, X2nd_Road_Class <int>,
## #   X2nd_Road_Number <int>, Pedestrian_Crossing.Human_Control <chr>,
## #   Pedestrian_Crossing.Physical_Facilities <chr>, Light_Conditions <chr>,
## #   Weather_Conditions <chr>, Road_Surface_Conditions <chr>,
## #   Special_Conditions_at_Site <chr>, Carriageway_Hazards <chr>,
## #   Urban_or_Rural_Area <int>,
## #   Did_Police_Officer_Attend_Scene_of_Accident <chr>,
## #   LSOA_of_Accident_Location <chr>, Year <int>
```

A head of accident data from 2009 to 2011

```
head(acc_09_11)
```

```
## # A tibble: 6 x 33
##   Accident_Index Location_Easting_O… Location_Northing… Longitude Latitude
##   <chr>                        <int>              <int>     <dbl>    <dbl>
## 1 200901BS70001                524910             180800    -0.201     51.5
## 2 200901BS70002                525050             181040    -0.199     51.5
## 3 200901BS70003                526490             177990    -0.180     51.5
## 4 200901BS70004                524800             180300    -0.203     51.5
## 5 200901BS70005                526930             177490    -0.173     51.5
## 6 200901BS70006                526060             178730    -0.186     51.5
## # ... with 28 more variables: Police_Force <int>, Accident_Severity <int>,
## #   Number_of_Vehicles <int>, Number_of_Casualties <int>, Date <chr>,
## #   Day_of_Week <int>, Time <chr>, Local_Authority_.District. <int>,
## #   Local_Authority_.Highway. <chr>, X1st_Road_Class <int>,
## #   X1st_Road_Number <int>, Road_Type <chr>, Speed_limit <int>,
## #   Junction_Detail <lgl>, Junction_Control <chr>, X2nd_Road_Class <int>,
## #   X2nd_Road_Number <int>, Pedestrian_Crossing.Human_Control <chr>,
```

```
## #   Pedestrian_Crossing.Physical_Facilities <chr>, Light_Conditions <chr>,
## #   Weather_Conditions <chr>, Road_Surface_Conditions <chr>,
## #   Special_Conditions_at_Site <chr>, Carriageway_Hazards <chr>,
## #   Urban_or_Rural_Area <int>,
## #   Did_Police_Officer_Attend_Scene_of_Accident <chr>,
## #   LSOA_of_Accident_Location <chr>, Year <int>
```

A head of accident data from 2012 to 2014

```
head(acc_12_14)
```

```
## # A tibble: 6 x 33
##   Accident_Index Location_Easting_O… Location_Northing… Longitude Latitude
##   <chr>                        <int>              <int>     <dbl>    <dbl>
## 1 201201BS70001                527200             178760    -0.169     51.5
## 2 201201BS70002                524930             181430    -0.201     51.5
## 3 201201BS70003                525860             178080    -0.189     51.5
## 4 201201BS70004                524980             181030    -0.200     51.5
## 5 201201BS70005                526170             179200    -0.184     51.5
## 6 201201BS70006                526090             177600    -0.185     51.5
## # ... with 28 more variables: Police_Force <int>, Accident_Severity <int>,
## #   Number_of_Vehicles <int>, Number_of_Casualties <int>, Date <chr>,
## #   Day_of_Week <int>, Time <chr>, Local_Authority_.District. <int>,
## #   Local_Authority_.Highway. <chr>, X1st_Road_Class <int>,
## #   X1st_Road_Number <int>, Road_Type <chr>, Speed_limit <int>,
## #   Junction_Detail <lgl>, Junction_Control <chr>, X2nd_Road_Class <int>,
## #   X2nd_Road_Number <int>, Pedestrian_Crossing.Human_Control <chr>,
## #   Pedestrian_Crossing.Physical_Facilities <chr>, Light_Conditions <chr>,
## #   Weather_Conditions <chr>, Road_Surface_Conditions <chr>,
## #   Special_Conditions_at_Site <chr>, Carriageway_Hazards <chr>,
## #   Urban_or_Rural_Area <int>,
## #   Did_Police_Officer_Attend_Scene_of_Accident <chr>,
## #   LSOA_of_Accident_Location <chr>, Year <int>
```

A head of traffic data

```
head(TrafficAADF)
```

```
## # A tibble: 6 x 29
##   AADFYear    CP Estimation_method Estimation_meth… Region  LocalAuthority
##      <int> <int> <chr>             <chr>            <chr>   <chr>
## 1     2000  6007 Counted           Manual count     Yorksh… Rotherham
## 2     2000  6009 Counted           Manual count     Yorksh… Leeds
## 3     2000  6035 Counted           Manual count     Yorksh… Doncaster
## 4     2000  6054 Counted           Manual count     Yorksh… Calderdale
## 5     2000  6055 Counted           Manual count     Yorksh… Leeds
## 6     2000  6056 Counted           Manual count     Yorksh… Wakefield
## # ... with 23 more variables: Road <chr>, RoadCategory <chr>,
## #   Easting <int>, Northing <int>, StartJunction <chr>, EndJunction <chr>,
## #   LinkLength_km <dbl>, LinkLength_miles <dbl>, PedalCycles <int>,
```

```
## #   Motorcycles <int>, CarsTaxis <int>, BusesCoaches <int>,
## #   LightGoodsVehicles <int>, V2AxleRigidHGV <int>, V3AxleRigidHGV <int>,
## #   V4or5AxleRigidHGV <int>, V3or4AxleArticHGV <int>,
## #   V5AxleArticHGV <int>, V6orMoreAxleArticHGV <int>, AllHGVs <int>,
## #   AllMotorVehicles <int>, Lat <dbl>, Lon <dbl>
```

7.1.5 We need to explore data

Before explore data, must know structure of data. So we can use some of function to do this like:

Summary () str() class()

We not need to repeat explore processes to others accidents files because all have the same structure,So we explore only one file.

Accident data

```
class(acc_05_07)
```

```
## [1] "tbl_df"     "tbl"       "data.frame"
```

```
str(acc_05_07)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    570011 obs. of  33 variables:
##  $ Accident_Index                : chr  "200501BS00001" "200501BS00002" "200501BS00003" "200501BS00004" ...
##  $ Location_Easting_OSGR         : int  525680 524170 524520 526900 528060 524770 524220 525890 527350
524550 ...
##  $ Location_Northing_OSGR        : int  178240 181650 182240 177530 179040 181160 180830 179710
177650 180810 ...
##  $ Longitude                     : num  -0.191 -0.212 -0.206 -0.174 -0.157 ...
##  $ Latitude                      : num  51.5 51.5 51.5 51.5 51.5 ...
##  $ Police_Force                  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Accident_Severity             : int  2 3 3 3 3 3 3 3 3 3 ...
##  $ Number_of_Vehicles            : int  1 1 2 1 1 2 2 1 2 2 ...
##  $ Number_of_Casualties          : int  1 1 1 1 1 1 1 2 2 5 ...
##  $ Date                          : chr  "04/01/2005" "05/01/2005" "06/01/2005" "07/01/2005" ...
##  $ Day_of_Week                   : int  3 4 5 6 2 3 5 6 7 7 ...
##  $ Time                          : chr  "17:42" "17:36" "00:15" "10:35" ...
##  $ Local_Authority_.District.    : int  12 12 12 12 12 12 12 12 12 12 ...
##  $ Local_Authority_.Highway.     : chr  "E09000020" "E09000020" "E09000020" "E09000020" ...
##  $ X1st_Road_Class               : int  3 4 5 3 6 6 5 3 3 4 ...
##  $ X1st_Road_Number              : int  3218 450 0 3220 0 0 0 315 3212 450 ...
##  $ Road_Type                     : chr  "Single carriageway" "Dual carriageway" "Single carriageway" "Single
carriageway" ...
##  $ Speed_limit                   : int  30 30 30 30 30 30 30 30 30 30 ...
##  $ Junction_Detail               : logi  NA NA NA NA NA NA ...
##  $ Junction_Control              : chr  "" "Automatic traffic signal" "" "" ...
##  $ X2nd_Road_Class               : int  -1 5 -1 -1 -1 -1 6 -1 4 5 ...
##  $ X2nd_Road_Number              : int  0 0 0 0 0 0 0 0 304 0 ...
```

```
## $ Pedestrian_Crossing.Human_Control        : chr  "None within 50 metres" "None within 50 metres" "None within
50 metres" "None within 50 metres" ...
## $ Pedestrian_Crossing.Physical_Facilities   : chr  "Zebra crossing" "Pedestrian phase at traffic signal junction" "No
physical crossing within 50 meters" "No physical crossing within 50 meters" ...
## $ Light_Conditions                  : chr  "Daylight: Street light present" "Darkness: Street lights present and lit"
"Darkness: Street lights present and lit" "Daylight: Street light present" ...
## $ Weather_Conditions                : chr  "Raining without high winds" "Fine without high winds" "Fine without
high winds" "Fine without high winds" ...
## $ Road_Surface_Conditions           : chr  "Wet/Damp" "Dry" "Dry" "Dry" ...
## $ Special_Conditions_at_Site        : chr  "None" "None" "None" "None" ...
## $ Carriageway_Hazards               : chr  "None" "None" "None" "None" ...
## $ Urban_or_Rural_Area               : int  1 1 1 1 1 1 1 1 1 1 ...
## $ Did_Police_Officer_Attend_Scene_of_Accident: chr  "Yes" "Yes" "Yes" "Yes" ...
## $ LSOA_of_Accident_Location         : chr  "E01002849" "E01002909" "E01002857" "E01002840" ...
## $ Year                              : int  2005 2005 2005 2005 2005 2005 2005 2005 2005 2005 ...
```

```
summary(acc_05_07)
```

```
## Accident_Index    Location_Easting_OSGR Location_Northing_OSGR
## Length:570011     Min.   : 64980        Min.   :  10520
## Class :character  1st Qu.:371020        1st Qu.: 178740
## Mode  :character  Median :435490        Median : 277385
##                   Mean   :435610        Mean   : 301725
##                   3rd Qu.:519230        3rd Qu.: 398280
##                   Max.   :655290        Max.   :1208800
##                   NA's   :101           NA's   :101
##    Longitude         Latitude      Police_Force   Accident_Severity
## Min.   :-7.5159   Min.   :49.91   Min.   : 1.00   Min.   :1.00
## 1st Qu.:-2.4346   1st Qu.:51.49   1st Qu.: 7.00   1st Qu.:3.00
## Median :-1.4704   Median :52.39   Median :31.00   Median :3.00
## Mean   :-1.4950   Mean   :52.60   Mean   :31.21   Mean   :2.84
## 3rd Qu.:-0.2678   3rd Qu.:53.48   3rd Qu.:46.00   3rd Qu.:3.00
## Max.   : 1.7583   Max.   :60.76   Max.   :98.00   Max.   :3.00
## NA's   :101       NA's   :101
## Number_of_Vehicles Number_of_Casualties     Date
## Min.   : 1.000     Min.   : 1.000       Length:570011
## 1st Qu.: 1.000     1st Qu.: 1.000       Class :character
## Median : 2.000     Median : 1.000       Mode  :character
## Mean   : 1.841     Mean   : 1.363
## 3rd Qu.: 2.000     3rd Qu.: 1.000
## Max.   :28.000     Max.   :68.000
##
##  Day_of_Week       Time          Local_Authority_.District.
## Min.   :1.000   Length:570011     Min.   :  1.0
## 1st Qu.:2.000   Class :character   1st Qu.:126.0
## Median :4.000   Mode  :character   Median :340.0
## Mean   :4.124                      Mean   :358.9
## 3rd Qu.:6.000                      3rd Qu.:536.0
## Max.   :7.000                      Max.   :941.0
##
```

```
## Local_Authority_.Highway. X1st_Road_Class X1st_Road_Number
## Length:570011          Min.   :1.000   Min.   : -1
## Class :character       1st Qu.:3.000   1st Qu.:   0
## Mode  :character       Median :4.000   Median : 112
##                        Mean   :4.107   Mean   :1002
##                        3rd Qu.:6.000   3rd Qu.: 694
##                        Max.   :6.000   Max.   :9999
##
##  Road_Type        Speed_limit    Junction_Detail Junction_Control
## Length:570011     Min.   :10.00   Mode:logical    Length:570011
## Class :character  1st Qu.:30.00   NA's:570011     Class :character
## Mode  :character  Median :30.00                   Mode  :character
##                   Mean   :39.79
##                   3rd Qu.:60.00
##                   Max.   :70.00
##
## X2nd_Road_Class  X2nd_Road_Number Pedestrian_Crossing.Human_Control
## Min.   :-1.000   Min.   : -1.0   Length:570011
## 1st Qu.:-1.000   1st Qu.:  0.0   Class :character
## Median : 3.000   Median :  0.0   Mode  :character
## Mean   : 2.589   Mean   : 379.4
## 3rd Qu.: 6.000   3rd Qu.:  0.0
## Max.   : 6.000   Max.   :9999.0
##
## Pedestrian_Crossing.Physical_Facilities Light_Conditions
## Length:570011                           Length:570011
## Class :character                        Class :character
## Mode  :character                        Mode  :character
##
##
##
##
## Weather_Conditions Road_Surface_Conditions Special_Conditions_at_Site
## Length:570011      Length:570011           Length:570011
## Class :character   Class :character        Class :character
## Mode  :character   Mode  :character        Mode  :character
##
##
##
##
## Carriageway_Hazards Urban_or_Rural_Area
## Length:570011       Min.   :1.000
## Class :character    1st Qu.:1.000
## Mode  :character    Median :1.000
##                     Mean   :1.367
##                     3rd Qu.:2.000
##                     Max.   :3.000
##
## Did_Police_Officer_Attend_Scene_of_Accident LSOA_of_Accident_Location
```

```
##  Length:570011                      Length:570011
##  Class :character                   Class :character
##  Mode  :character                   Mode  :character
##
##
##
##
##      Year
##  Min.   :2005
##  1st Qu.:2005
##  Median :2006
##  Mean   :2006
##  3rd Qu.:2007
##  Max.   :2007
##
```

Traffic data

```
class(TrafficAADF)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
str(TrafficAADF)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    275385 obs. of  29 variables:
##  $ AADFYear               : int  2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 ...
##  $ CP                     : int  6007 6009 6035 6054 6055 6056 6057 6063 6066 6088 ...
##  $ Estimation_method      : chr  "Counted" "Counted" "Counted" "Counted" ...
##  $ Estimation_method_detailed: chr  "Manual count" "Manual count" "Manual count" "Manual count" ...
##  $ Region                 : chr  "Yorkshire and the Humber" "Yorkshire and the Humber" "Yorkshire and the Humber"
"Yorkshire and the Humber" ...
##  $ LocalAuthority         : chr  "Rotherham" "Leeds" "Doncaster" "Calderdale" ...
##  $ Road                   : chr  "M1" "M621" "M18" "M62" ...
##  $ RoadCategory           : chr  "TM" "TM" "TM" "TM" ...
##  $ Easting                : int  446000 432150 466400 404000 426000 446000 474600 426500 454000 439926 ...
##  $ Northing               : int  389300 429500 407900 416600 426200 424000 427300 429600 403800 463500 ...
##  $ StartJunction          : chr  "M18 spur" "7" "4" "22" ...
##  $ EndJunction            : chr  "33" "M1" "5" "LA Boundary" ...
##  $ LinkLength_km          : num  3.8 1.9 6.5 6 4.9 5.2 0.7 4 5.6 8.2 ...
##  $ LinkLength_miles       : num  2.36 1.18 4.04 3.73 3.04 3.23 0.43 2.49 3.48 5.1 ...
##  $ PedalCycles            : int  0 0 0 0 0 0 1 0 0 0 ...
##  $ Motorcycles            : int  168 196 25 130 229 138 25 242 190 148 ...
##  $ CarsTaxis              : int  81418 52479 26170 58960 68722 41154 9075 40973 40107 41853 ...
##  $ BusesCoaches           : int  440 291 129 319 282 180 49 151 330 337 ...
##  $ LightGoodsVehicles     : int  13950 8078 5017 11241 11818 7684 1578 6848 6892 7152 ...
##  $ V2AxleRigidHGV         : int  5351 1293 1952 3121 4030 3300 681 1037 2386 3350 ...
##  $ V3AxleRigidHGV         : int  656 272 257 508 624 493 90 183 410 393 ...
##  $ V4or5AxleRigidHGV      : int  711 287 239 354 423 601 101 99 550 319 ...
##  $ V3or4AxleArticHGV      : int  1588 628 745 1786 1947 1261 325 314 1419 1982 ...
##  $ V5AxleArticHGV         : int  4715 1494 3695 5902 6669 4723 716 854 4709 6628 ...
##  $ V6orMoreAxleArticHGV   : int  2149 921 2000 3438 3743 3337 725 408 2569 4204 ...
```

```
## $ AllHGVs               : int  15170 4895 8888 15109 17436 13715 2638 2895 12043 16876 ...
## $ AllMotorVehicles      : int  111146 65939 40229 85759 98487 62871 13365 51109 59562 66366 ...
## $ Lat                   : num  53.4 53.8 53.6 53.6 53.7 ...
## $ Lon                   : num  -1.31 -1.514 -0.999 -1.941 -1.607 ...
```

```
summary(TrafficAADF)
```

```
##    AADFYear        CP       Estimation_method
## Min.   :2000   Min.   :   60   Length:275385
## 1st Qu.:2004   1st Qu.:26218   Class :character
## Median :2008   Median :47252   Mode  :character
## Mean   :2008   Mean   :47277
## 3rd Qu.:2012   3rd Qu.:74392
## Max.   :2016   Max.   :99967
## Estimation_method_detailed    Region        LocalAuthority
## Length:275385              Length:275385      Length:275385
## Class :character           Class :character   Class :character
## Mode  :character           Mode  :character   Mode  :character
##
##
##
##     Road         RoadCategory         Easting         Northing
## Length:275385      Length:275385      Min.   : 69987   Min.   :  76250
## Class :character   Class :character   1st Qu.:352970   1st Qu.: 192120
## Mode  :character   Mode  :character   Median :430000   Median : 324000
##                                       Mean   :425675   Mean   : 356183
##                                       3rd Qu.:510600   3rd Qu.: 429600
##                                       Max.   :655040   Max.   :1205400
## StartJunction      EndJunction        LinkLength_km    LinkLength_miles
## Length:275385      Length:275385      Min.   : 0.070   Min.   : 0.040
## Class :character   Class :character   1st Qu.: 0.600   1st Qu.: 0.370
## Mode  :character   Mode  :character   Median : 1.400   Median : 0.870
##                                       Mean   : 2.757   Mean   : 1.713
##                                       3rd Qu.: 3.200   3rd Qu.: 1.990
##                                       Max.   :55.500   Max.   :34.490
##  PedalCycles      Motorcycles      CarsTaxis      BusesCoaches
## Min.   :   0.0   Min.   :  0.0   Min.   :    0   Min.   :   0.0
## 1st Qu.:   5.0   1st Qu.: 58.0   1st Qu.: 7125   1st Qu.:  60.0
## Median :  26.0   Median : 113.0   Median : 12255   Median :  127.0
## Mean   :  123.3   Mean   : 222.1   Mean   : 16813   Mean   :  248.7
## 3rd Qu.:  83.0   3rd Qu.: 214.0   3rd Qu.: 19951   3rd Qu.:  280.0
## Max.   :18629.0   Max.   :9815.0   Max.   :207133   Max.   :11359.0
## LightGoodsVehicles V2AxleRigidHGV    V3AxleRigidHGV    V4or5AxleRigidHGV
## Min.   :   0     Min.   :   0.0   Min.   :  0.00   Min.   :  0.00
## 1st Qu.: 1070     1st Qu.:  148.0   1st Qu.:  24.00   1st Qu.:  15.00
## Median : 1792     Median :  273.0   Median :  46.00   Median :  41.00
## Mean   : 2613     Mean   :  489.4   Mean   :  83.55   Mean   :  88.92
## 3rd Qu.: 2978     3rd Qu.:  517.0   3rd Qu.:  91.00   3rd Qu.: 100.00
## Max.   :38449     Max.   :10942.0   Max.   :5968.00   Max.   :3684.00
## V3or4AxleArticHGV V5AxleArticHGV    V6orMoreAxleArticHGV    AllHGVs
```

```
##  Min.   :  0.00   Min.   :   0.0   Min.   :   0.0     Min.   :   0
##  1st Qu.:   7.00   1st Qu.:   9.0   1st Qu.:  10.0     1st Qu.:  252
##  Median :  20.00   Median :  34.0   Median :  37.0     Median :  489
##  Mean   :  74.25   Mean   :  247.9   Mean   :  271.7     Mean   : 1256
##  3rd Qu.:  56.00   3rd Qu.:  125.0   3rd Qu.:  137.0     3rd Qu.: 1041
##  Max.   :3949.00   Max.   :11034.0   Max.   :13758.0     Max.   :27095
##  AllMotorVehicles      Lat            Lon
##  Min.   :    0   Min.   :50.58   Min.   :-7.4427
##  1st Qu.:  8941   1st Qu.:51.61   1st Qu.:-2.7115
##  Median : 15143   Median :52.81   Median :-1.5495
##  Mean   : 21153   Mean   :53.09   Mean   :-1.6546
##  3rd Qu.: 24660   3rd Qu.:53.76   3rd Qu.:-0.3898
##  Max.   :262842   Max.   :60.73   Max.   : 1.7546
```

7.1.6 Now our data need to clean duplicates rows and remove missing values

We want to check if traffic data and accidents data have duplicated data or not. By using duplicated function and sum.

```
sum(duplicated(acc_05_07))
```

```
## [1] 3
```

```
sum(duplicated(acc_09_11))
```

```
## [1] 5
```

```
sum(duplicated(acc_12_14))
```

```
## [1] 34147
```

```
sum(duplicated(TrafficAADF))
```

```
## [1] 0
```

We found duplicated data just in accidents data files and with very very low percentage, so we remove it using unique() function then resave it in same object.

```
acc_05_07 <- unique(acc_05_07)
acc_09_11 <- unique(acc_09_11)
acc_12_14 <- unique(acc_12_14)
```

7.1.7 Next step is remove rows which include missing value from data.

Check if data have missing values:

```
sum(is.na(acc_05_07))
```

```
## [1] 570412
```

```
sum(is.na(acc_09_11))
```

```
## [1] 469437
```

```
sum(is.na(acc_12_14))
```

```
## [1] 430550
```

```
sum(is.na(TrafficAADF))
```

```
## [1] 0
```

Missing value just found in accident data files na.omit() use to remove all missing values in data frame if found, but we not remove it now ,because data have junction_detail column have a lot missing value if remove it, we will remove all data, we will remove missing values in next chapter.

7.1.8 The accident data divided into three files, we will combine data to can visualize.

We use rbind() function to combine accident data files, where all accident files have the same strucure.

```
AllAccidents <- rbind(acc_05_07,acc_09_11,acc_12_14)
head(AllAccidents)
```

```
## # A tibble: 6 x 33
##   Accident_Index Location_Easting_O… Location_Northing… Longitude Latitude
##   <chr>                        <int>              <int>     <dbl>    <dbl>
## 1 200501BS00001               525680             178240    -0.191     51.5
## 2 200501BS00002               524170             181650    -0.212     51.5
## 3 200501BS00003               524520             182240    -0.206     51.5
## 4 200501BS00004               526900             177530    -0.174     51.5
## 5 200501BS00005               528060             179040    -0.157     51.5
## 6 200501BS00006               524770             181160    -0.203     51.5
## # ... with 28 more variables: Police_Force <int>, Accident_Severity <int>,
## #   Number_of_Vehicles <int>, Number_of_Casualties <int>, Date <chr>,
## #   Day_of_Week <int>, Time <chr>, Local_Authority_.District. <int>,
## #   Local_Authority_.Highway. <chr>, X1st_Road_Class <int>,
## #   X1st_Road_Number <int>, Road_Type <chr>, Speed_limit <int>,
## #   Junction_Detail <lgl>, Junction_Control <chr>, X2nd_Road_Class <int>,
## #   X2nd_Road_Number <int>, Pedestrian_Crossing.Human_Control <chr>,
## #   Pedestrian_Crossing.Physical_Facilities <chr>, Light_Conditions <chr>,
## #   Weather_Conditions <chr>, Road_Surface_Conditions <chr>,
## #   Special_Conditions_at_Site <chr>, Carriageway_Hazards <chr>,
## #   Urban_or_Rural_Area <int>,
```

```
## #   Did_Police_Officer_Attend_Scene_of_Accident <chr>,
## #   LSOA_of_Accident_Location <chr>, Year <int>
```

```
summary(AllAccidents)
```

```
##  Accident_Index     Location_Easting_OSGR Location_Northing_OSGR
##  Length:1469995     Min.   : 64950        Min.   :  10290
##  Class :character   1st Qu.:375750        1st Qu.: 178008
##  Mode  :character   Median :440930        Median : 265340
##                     Mean   :439897        Mean   : 298675
##                     3rd Qu.:523290        3rd Qu.: 396600
##                     Max.   :655370        Max.   :1208800
##                     NA's   :101           NA's   :101
##    Longitude          Latitude      Police_Force   Accident_Severity
##  Min.   :-7.5162   Min.   :49.91   Min.   : 1.00   Min.   :1.000
##  1st Qu.:-2.3637   1st Qu.:51.49   1st Qu.: 7.00   1st Qu.:3.000
##  Median :-1.3916   Median :52.28   Median :31.00   Median :3.000
##  Mean   :-1.4326   Mean   :52.58   Mean   :30.78   Mean   :2.839
##  3rd Qu.:-0.2185   3rd Qu.:53.46   3rd Qu.:46.00   3rd Qu.:3.000
##  Max.   : 1.7594   Max.   :60.76   Max.   :98.00   Max.   :3.000
##  NA's   :101       NA's   :101
##  Number_of_Vehicles Number_of_Casualties     Date
##  Min.   : 1.000     Min.   : 1.000       Length:1469995
##  1st Qu.: 1.000     1st Qu.: 1.000       Class :character
##  Median : 2.000     Median : 1.000       Mode  :character
##  Mean   : 1.832     Mean   : 1.351
##  3rd Qu.: 2.000     3rd Qu.: 1.000
##  Max.   :67.000     Max.   :93.000
##
##   Day_of_Week        Time          Local_Authority_.District.
##  Min.   :1.000   Length:1469995     Min.   :  1.0
##  1st Qu.:2.000   Class :character   1st Qu.:122.0
##  Median :4.000   Mode  :character   Median :328.0
##  Mean   :4.119                      Mean   :353.6
##  3rd Qu.:6.000                      3rd Qu.:532.0
##  Max.   :7.000                      Max.   :941.0
##
##  Local_Authority_.Highway. X1st_Road_Class X1st_Road_Number
##  Length:1469995            Min.   :1.00    Min.   :  -1
##  Class :character          1st Qu.:3.00    1st Qu.:   0
##  Mode  :character          Median :4.00    Median : 129
##                            Mean   :4.09    Mean   :1009
##                            3rd Qu.:6.00    3rd Qu.: 726
##                            Max.   :6.00    Max.   :9999
##
```

```
class(AllAccidents)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

7.1.9 We add columns for traffic value which have annual traffic volume in each road. So multiple in 365(number of day in year).

And add new column which have sum of all vehicles in day.

```
TrafficAADF <- mutate(TrafficAADF,
                      PedalCyclesVolume = PedalCycles*LinkLength_miles*365,
                      MotorcyclesVolume = Motorcycles*LinkLength_miles*365,
                      BusesCoachesVolume = BusesCoaches*LinkLength_miles*365,
                      LightGoodsVehiclesVolume = LightGoodsVehicles*LinkLength_miles*365,
                      V2AxleRigidHGVVolume = V2AxleRigidHGV*LinkLength_miles*365,
                      V3AxleRigidHGVVolume = V3AxleRigidHGV*LinkLength_miles*365,
                      V3or4AxleArticHGVVolume = V3or4AxleArticHGV*LinkLength_miles*365,
                      V4or5AxleRigidHGVVolume = V4or5AxleRigidHGV*LinkLength_miles*365,
                      CarsTaxisVolume = CarsTaxis*LinkLength_miles*365,
                      V5AxleArticHGVVolume = V5AxleArticHGV*LinkLength_miles*365,
                      V6orMoreAxleArticHGVVolume = V6orMoreAxleArticHGV*LinkLength_miles*365,
                      allVehicles=AllMotorVehicles+PedalCycles,
                      allVolume = V6orMoreAxleArticHGVVolume + V5AxleArticHGVVolume+
                       CarsTaxisVolume+V4or5AxleRigidHGVVolume+V3or4AxleArticHGVVolume+
                       V3AxleRigidHGVVolume+V2AxleRigidHGVVolume+LightGoodsVehiclesVolume+
                       BusesCoachesVolume+MotorcyclesVolume+PedalCyclesVolume
                     )
```

7.1.10 we edit the road category name errors

```
TrafficAADF$RoadCategory[TrafficAADF$RoadCategory == "Pu"] <- "PU"
TrafficAADF$RoadCategory[TrafficAADF$RoadCategory == "Tu"] <- "TU"
```

7.1.10.1 We transform data in day of week in accidents file form number to character. Before:

```
head(AllAccidents$Day_of_Week)
```

```
## [1] 3 4 5 6 2 3
```

Process:

```
AllAccidents$Day_of_Week[AllAccidents$Day_of_Week== 1]<-"Monday"
AllAccidents$Day_of_Week[AllAccidents$Day_of_Week== 2]<-"Tuesday"
AllAccidents$Day_of_Week[AllAccidents$Day_of_Week== 3]<-"Wendsday"
AllAccidents$Day_of_Week[AllAccidents$Day_of_Week== 4]<-"Thursday"
AllAccidents$Day_of_Week[AllAccidents$Day_of_Week== 5]<-"Friday"
AllAccidents$Day_of_Week[AllAccidents$Day_of_Week== 6]<-"Saturday"
AllAccidents$Day_of_Week[AllAccidents$Day_of_Week== 7]<-"Sunday"
```

After:

```
head(AllAccidents$Day_of_Week)
```

```
## [1] "Wendsday" "Thursday" "Friday"   "Saturday" "Tuesday"  "Wendsday"
```

7.1.10.2 We will rename the accident Urban_or_Rural_Area

Before:

```
head(AllAccidents$Urban_or_Rural_Area)
```

```
## [1] 1 1 1 1 1 1
```

Process:

```
AllAccidents$Urban_or_Rural_Area[AllAccidents$Urban_or_Rural_Area== 1]<-"Rural"
AllAccidents$Urban_or_Rural_Area[AllAccidents$Urban_or_Rural_Area== 2]<-"Urban"
AllAccidents$Urban_or_Rural_Area[AllAccidents$Urban_or_Rural_Area== 3]<-"missing values"
```

After:

```
head(AllAccidents$Urban_or_Rural_Area)
```

```
## [1] "Rural" "Rural" "Rural" "Rural" "Rural" "Rural"
```

To make x axe vertical

```
p<-theme_bw()+theme(axis.text.x = element_text(angle=90, hjust=1),strip.text.y = element_text(angle = 0))
```

Second proplem step in my project:

7.2 Visualization Data Process

Visualization considers the main processes to solve the problem. Where we start to search the relation and causes between variables and get results.

We start to solve data problems which be writen in before

7.2 .1 first problem

Problem*:* How have Rural and Urban areas differed (See RoadCategory column)

Load package to work on data

```
library(dplyr)
library(tidyr)
library(ggplot2)
```

7.2 .1 .1 Selected the needed columns from accidents object

```
accEighthGoal <- AllAccidents %>%
  select(Accident_Index,Urban_or_Rural_Area)
```

7.2 .1 .2 Select the needed columns from traffic object

```
trafficEighthGoal <- TrafficAADF %>%
  select(AADFYear,Region,LocalAuthority,Road,RoadCategory,PedalCyclesVolume,
      MotorcyclesVolume,BusesCoachesVolume,
      LightGoodsVehiclesVolume,
      V2AxleRigidHGVVolume,V3AxleRigidHGVVolume,
      CarsTaxisVolume,V3or4AxleArticHGVVolume,
      V4or5AxleRigidHGVVolume,V5AxleArticHGVVolume,
      V6orMoreAxleArticHGVVolume,
      AllHGVs,AllMotorVehicles,allVolume,allVehicles)
```

7.2 .1 .3 Check if have missing value

```
sum(is.na(trafficEighthGoal))
```

```
## [1] 0
```

```
sum(is.na(accEighthGoal))
```

```
## [1] 0
```

We not found any missing value.

7.2 .1 .4 See where the accident be more in rural or urban

```
g<- ggplot(accEighthGoal,aes(Urban_or_Rural_Area,fill = factor(Urban_or_Rural_Area)))
g+geom_bar()+labs(title = "the accident rate in Rural and Urban")
```

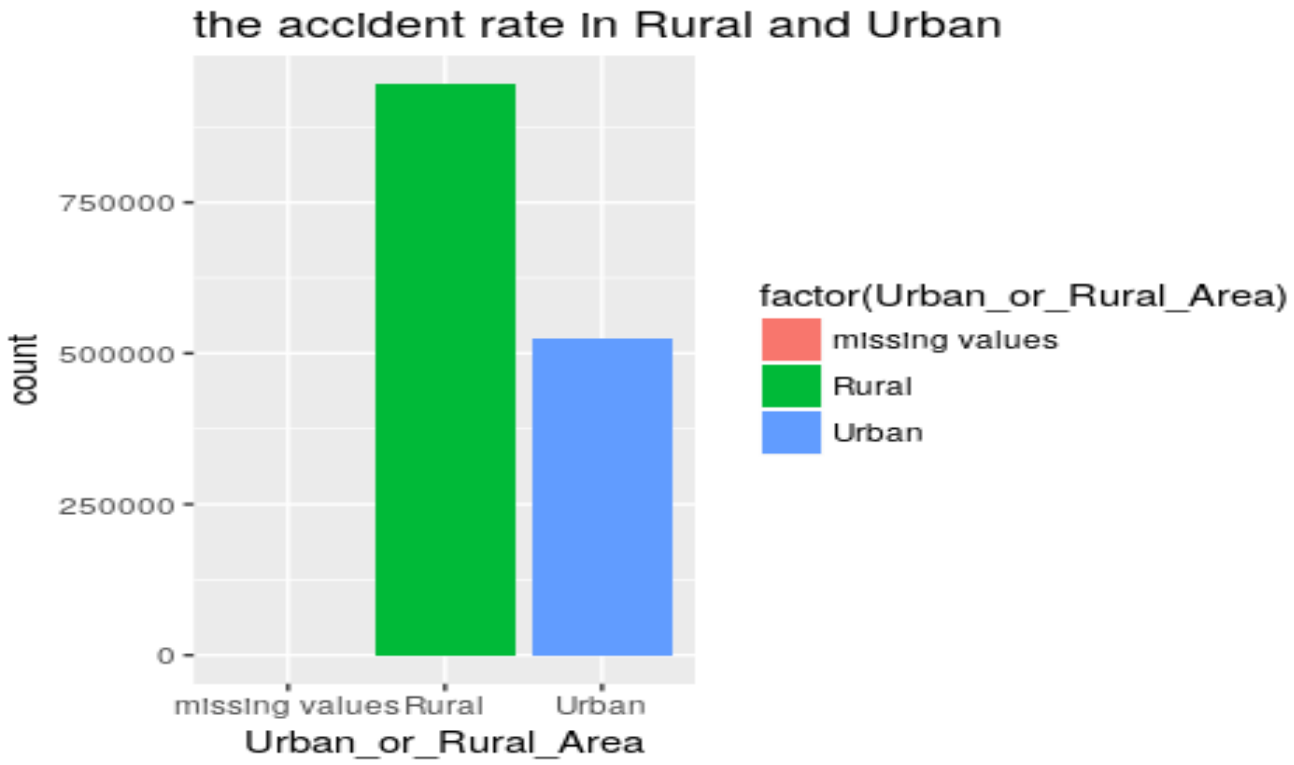


FIGURE7.1:THE ACCIDENT RATE IN RURAL AND URBAN

THE ACCIDENT IN RURAL IS MORE THAN IN URBAN.

7.2 .1 .5 See which road category be more flow

```
g <- ggplot(trafficEighthGoal,aes(RoadCategory,allVolume,colour = "red"))
g+geom_point()+theme_bw()+labs(title = "flow for each road category")
```

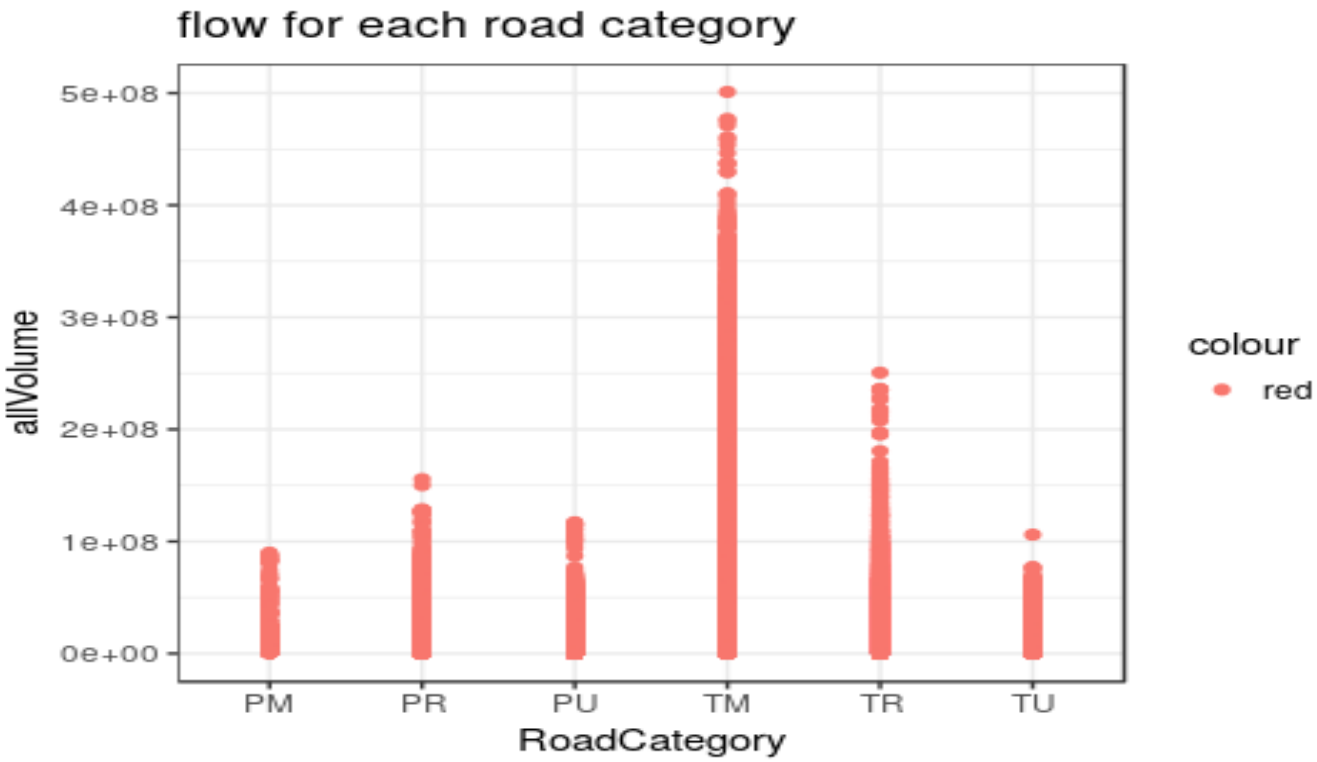


FIGURE7.2: FLOW FOR EACH ROAD CATEGORY

We see that in the first place is TM, It is very busy road category, the order of others roads of category as you see.

7.2 .1 .6 Filter data to get the rural and urban roads for road category

```
UrbanRural <- filter(trafficEighthGoal,RoadCategory == "TR" | RoadCategory == "PR"
              | RoadCategory == "PU" | RoadCategory == "TU") %>%
 group_by(AADFYear,RoadCategory)
```

We summarize data to get average of traffic volume in Rural and Urban.

```
average_category <- summarise(UrbanRural,meanallVolume = mean(allVolume))
head(average_category)
```

```
## # A tibble: 6 x 3
## # Groups:   AADFYear [2]
##   AADFYear RoadCategory meanallVolume
##      <int> <chr>              <dbl>
## 1     2000 PR              7360678.
## 2     2000 PU              5256765.
## 3     2000 TR             17022188.
## 4     2000 TU              9818433.
## 5     2001 PR              7519632.
## 6     2001 PU              5592029.
```

```
g<-ggplot(average_category,aes(AADFYear,meanallVolume,color = RoadCategory))
g+geom_point()+theme_bw()+geom_smooth(method = "loess",se =
F)+scale_x_discrete(limits=(2000:2016))+labs(title = "the rural and urban roads for road category")
```

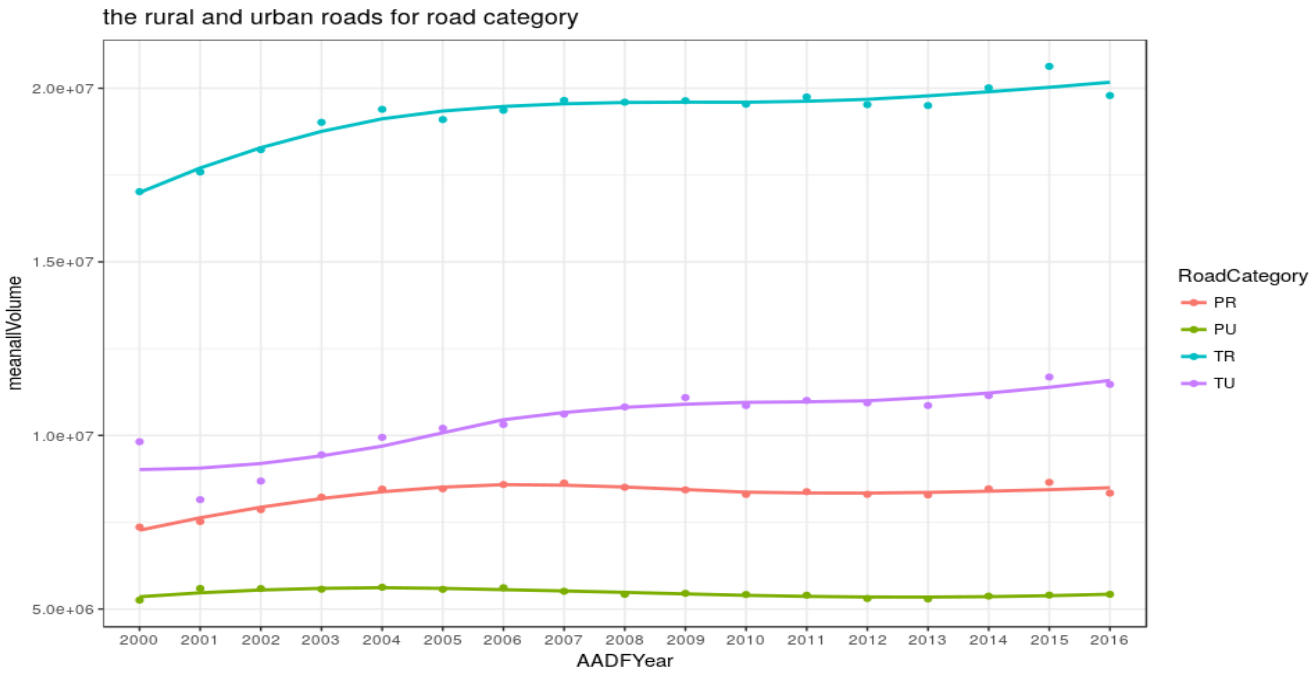


FIGURE7.3: THE RURAL AND URBAN ROADS FOR ROAD CATEGORY

We see that the average of traffic volume in Rural is be more than in Urban.

7.2 .1 .7 We study the average of each vehicles in over years in road category

```
g <- ggplot(average_category,aes(AADFYear,meanallVolume,color = AADFYear))
g+geom_point()+geom_smooth(method = "lm",se = F)+facet_grid(.~RoadCategory)+theme_bw()+labs(title
= "the average of each vehicles in over years in road category")
```

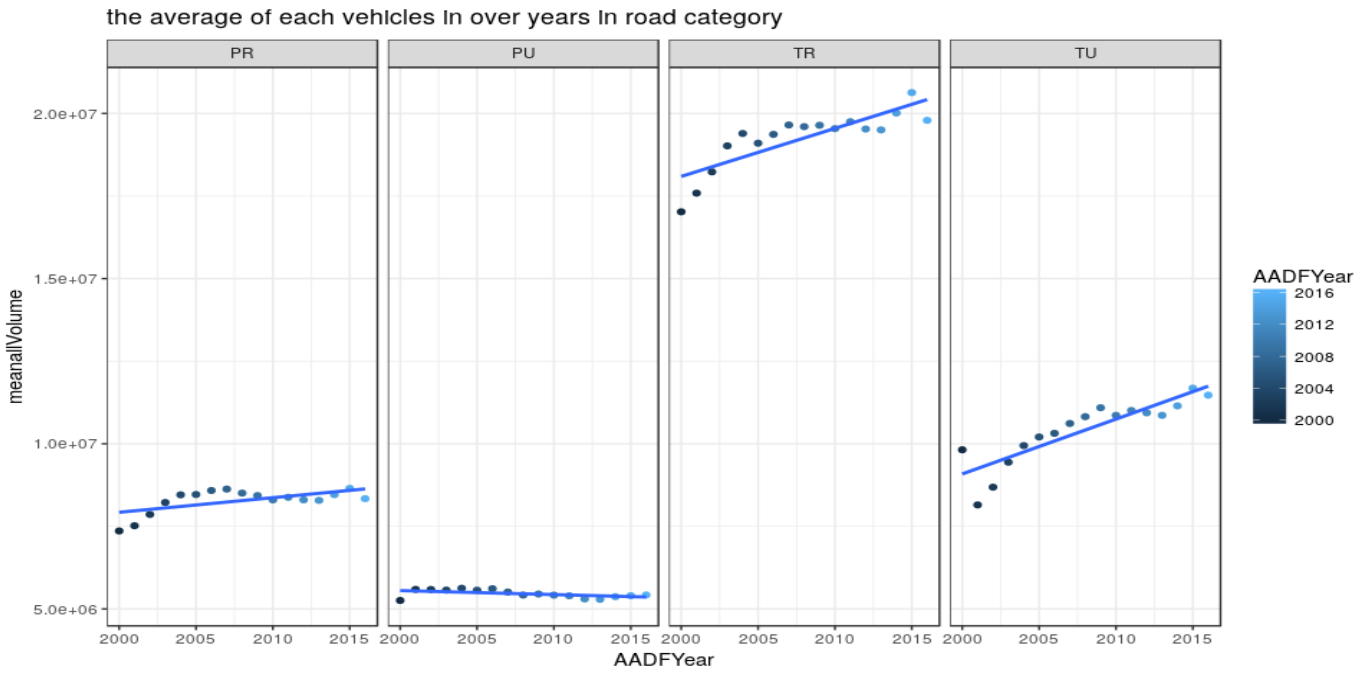


FIGURE7.4: THE AVERAGE OF EACH VEHICLES IN OVER YEARS IN ROAD CATEGORY

We see that the average of vehicles in Rural is be more than in Urban.

7.2 .1 .8 Calculate the global average of all vehicles to use it as standard

```
global_average <- mean(TrafficAADF$allVolume)
global_average
```

```
## [1] 11005156
```

7.2 .1 .9 See the flow of each vehicle in Urban and Rural for whole road category

7.2 .1 .9.1 Lets start with pedal cycles

```
g <- ggplot(UrbanRural,aes(RoadCategory,PedalCyclesVolume,fill = factor (RoadCategory)))
g+geom_bar(stat ="identity")+facet_grid(.~AADFYear)+theme_bw()+geom_hline(yintercept =
global_average )+labs(title = "the flow of pedal cycles in urban and rural for whole road category")
```

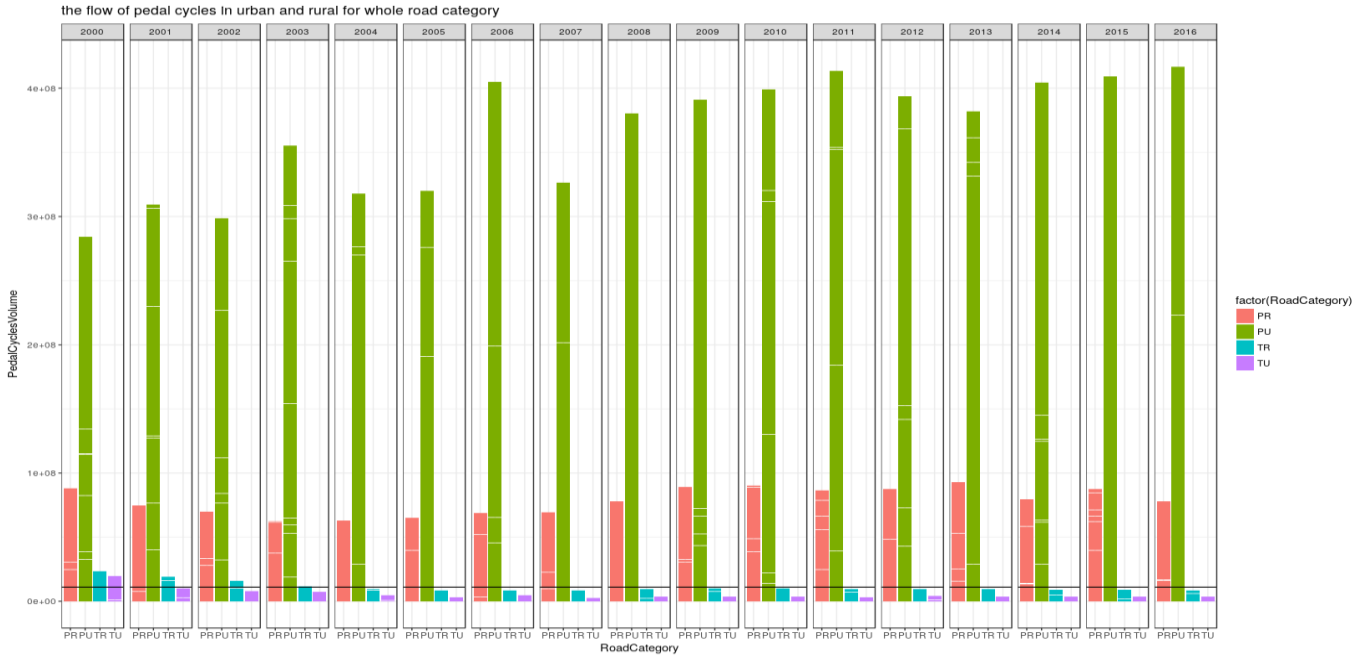


FIGURE7.5:THE FLOW OF PEDAL CYCLES IN URBAN AND RURAL FOR WHOLE ROAD CATEGORY

We see that the pedal cycles pass the global average, see that the average of pedal cycle in PU is very large in Urban than Rural Usage is increasing over years.

7.2 .1 .9.2 motor cycles

```
g <- ggplot(UrbanRural,aes(RoadCategory,MotorcyclesVolume,fill =factor( RoadCategory)))
g+geom_bar(stat ="identity")+facet_grid(.~AADFYear)+theme_bw()+geom_hline(yintercept =
global_average )+labs(title = "the flow of motor cycles in urban and rural for whole road category")
```

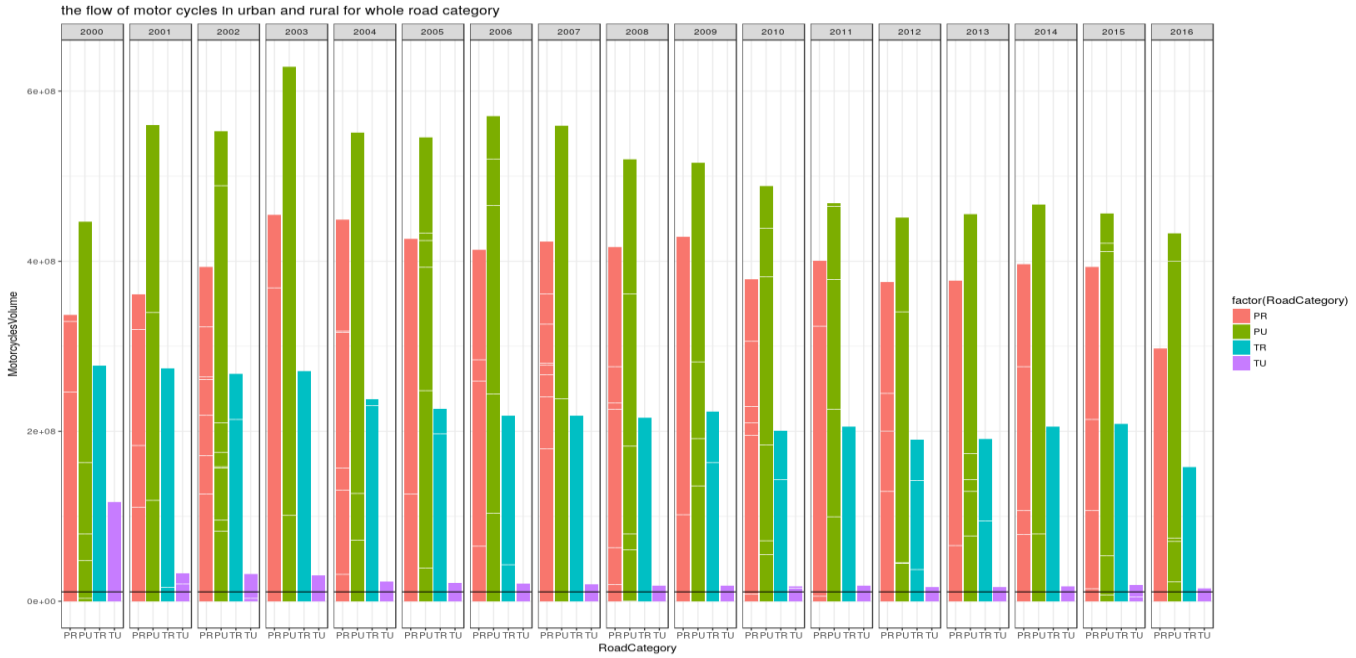


FIGURE7.6: THE FLOW OF MOTOR CYCLES IN URBAN AND RURAL FOR WHOLE ROAD CATEGORY

We see that the average of motor cycle is decreasing over years.

Pass the global average.

Used in PU road category is large.

7.2 .1 .9.3 buses coaches

```
g <- ggplot(UrbanRural,aes(RoadCategory,BusesCoachesVolume,fill = factor( RoadCategory)))
g+geom_bar(stat ="identity")+facet_grid(.~AADFYear)+theme_bw()+geom_hline(yintercept =
global_average )+labs(title = "the flow of buses coaches in urban and rural for whole road category")
```

We see that the average of buses coaches is still stand over years almost in all road categories.

Pass the global average.

Used in PU road category is large.

7.2 .1 .9.4 Light goods

```
g <- ggplot(UrbanRural,aes(RoadCategory,LightGoodsVehiclesVolume,fill = factor(RoadCategory)))
g+geom_bar(stat ="identity")+facet_grid(.~AADFYear)+theme_bw()+geom_hline(yintercept =
global_average )+labs(title = "the flow of light goods in urban and rural for whole road category")
```

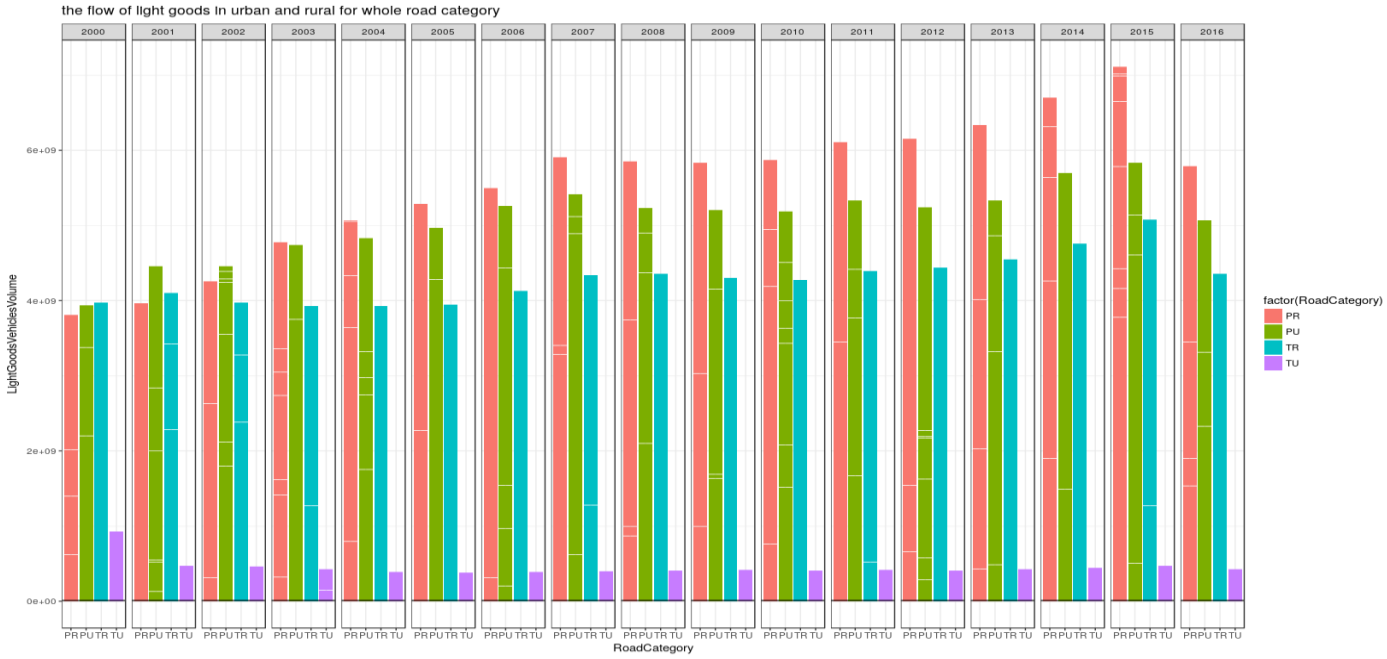


FIGURE7.7:THE FLOW OF LIGHT GOODS IN URBAN AND RURAL FOR WHOLE ROAD CATEGORY

All road categories overcome the global average.

The light goods used more in rural roads spechially PR roads.

The useage is still stand in almost over year in TU roads.

In PR road the useage is increased over years.

In other roads, the useage is still stand.

7.2 .1 .9.5 V2AxleRigidHGV

```
g <- ggplot(UrbanRural,aes(RoadCategory,V2AxleRigidHGVVolume,fill =factor( RoadCategory)))
g+geom_bar(stat ="identity")+facet_grid(.~AADFYear)+theme_bw()+geom_hline(yintercept =
global_average )+labs(title = "the flow of V2AxleRigidHGV in urban and rural for whole road category")
```

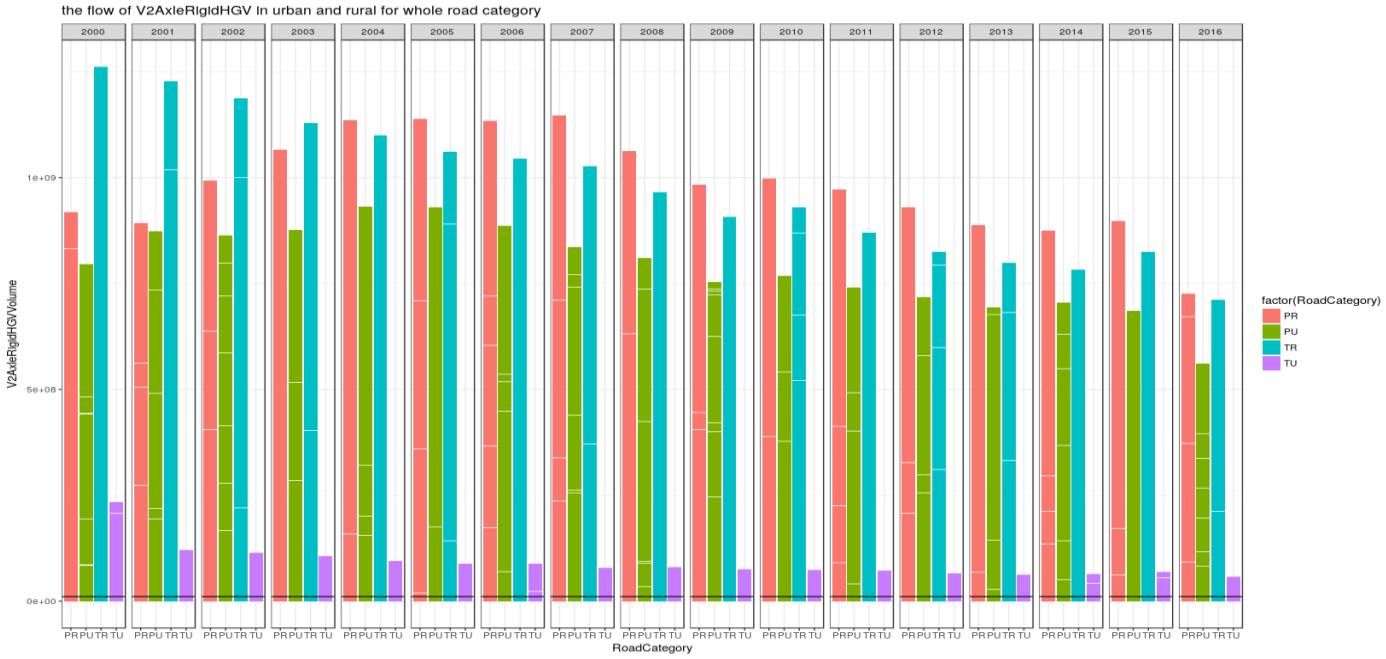


FIGURE7.8: THE FLOW OF V2AXLERIGIDHGV IN URBAN AND RURAL FOR WHOLE ROAD CATEGORY

All roads pass the global average.

Useage is large in TR road category from 2000 : 2003,then the TR is large from 2004: 2016.

The useage decreases over years in all road category except TU roads is still stand.

7.2 .1 .9.6 V3AxleRigidHGV

```
g <- ggplot(UrbanRural,aes(RoadCategory,V3AxleRigidHGVVolume,fill = factor(RoadCategory)))
g+geom_bar(stat ="identity")+facet_grid(.~AADFYear)+theme_bw()+geom_hline(yintercept =
global_average )+labs(title = "the flow of V3AxleRigidHGV in urban and rural for whole road category")
```

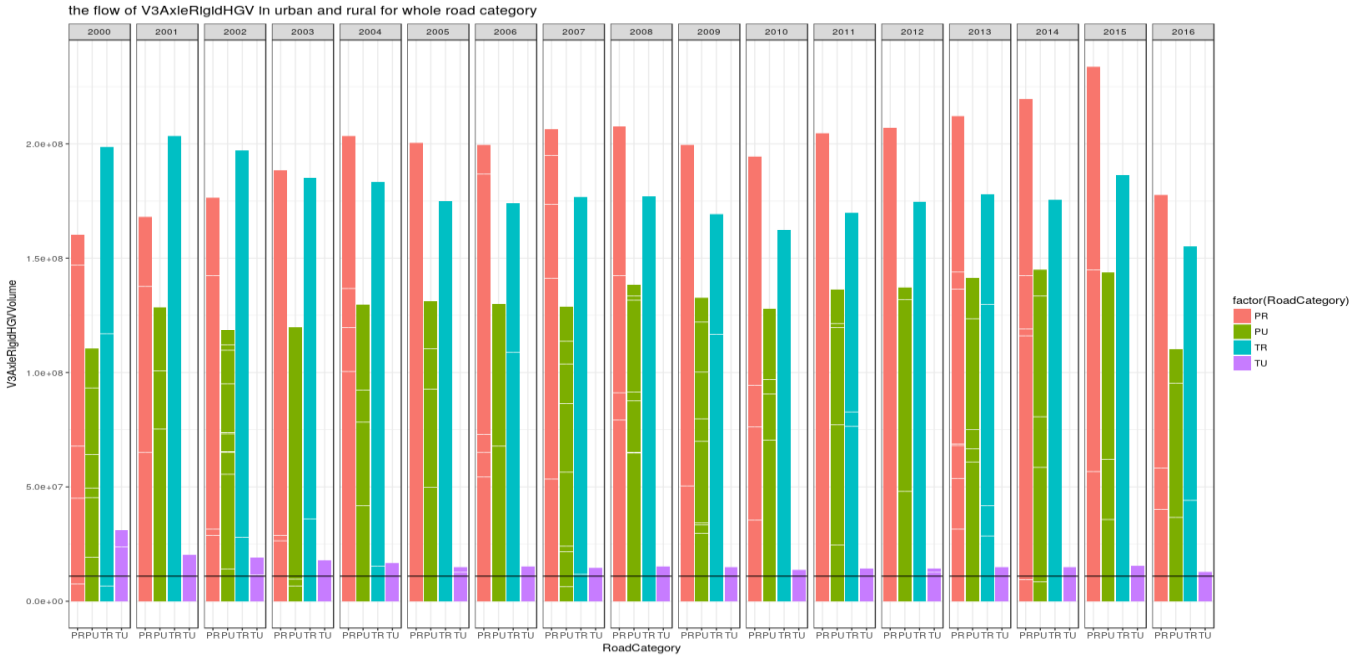


FIGURE7.9: THE FLOW OF V3AXLERIGIDHGV IN URBAN AND RURAL FOR WHOLE ROAD CATEGORY

All roads pass the global average.

Still stand over years almost from 2000 : 2002 the TR roads is the busiest roads,then form 2003:2016 the PR is the busiest roads.

7.2 .1 .9.7 CarsTaxis

```
g <- ggplot(UrbanRural,aes(RoadCategory,CarsTaxisVolume,fill = factor(RoadCategory)))
g+geom_bar(stat ="identity")+facet_grid(.~AADFYear)+theme_bw()+geom_hline(yintercept =
global_average )+labs(title = "the flow of car taxis in urban and rural for whole road category")
```

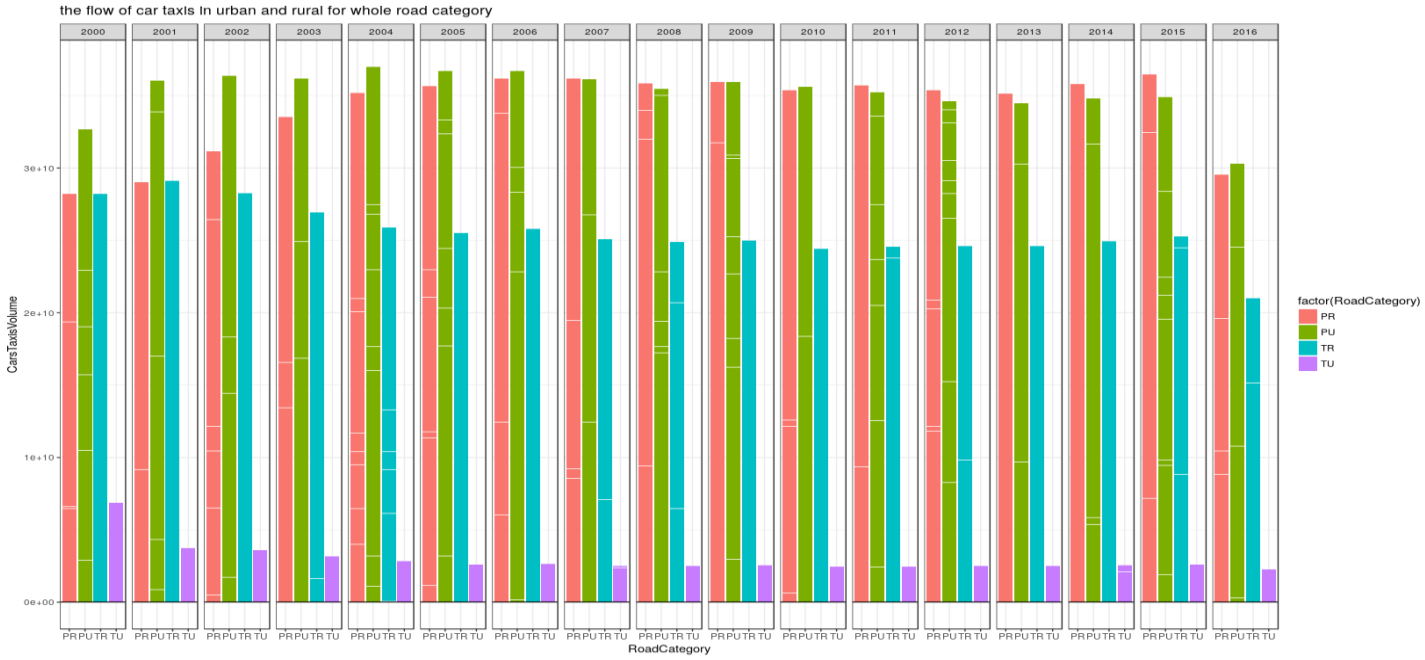


FIGURE7 10: THE FLOW OF CAR TAXIS IN URBAN AND RURAL FOR WHOLE ROAD CATEGORY

All road categories are still stand over years and decrease in year of 2016.

7.2 .1 .9.8 V3or4AxleArticHGV

```
g <- ggplot(UrbanRural,aes(RoadCategory,V3or4AxleArticHGVVolume,fill = factor(RoadCategory)))
g+geom_bar(stat ="identity")+facet_grid(.~AADFYear)+theme_bw()+geom_hline(yintercept =
global_average )+labs(title = "the flow of V3or4AxleRigidHGV in urban and rural for whole road category")
```

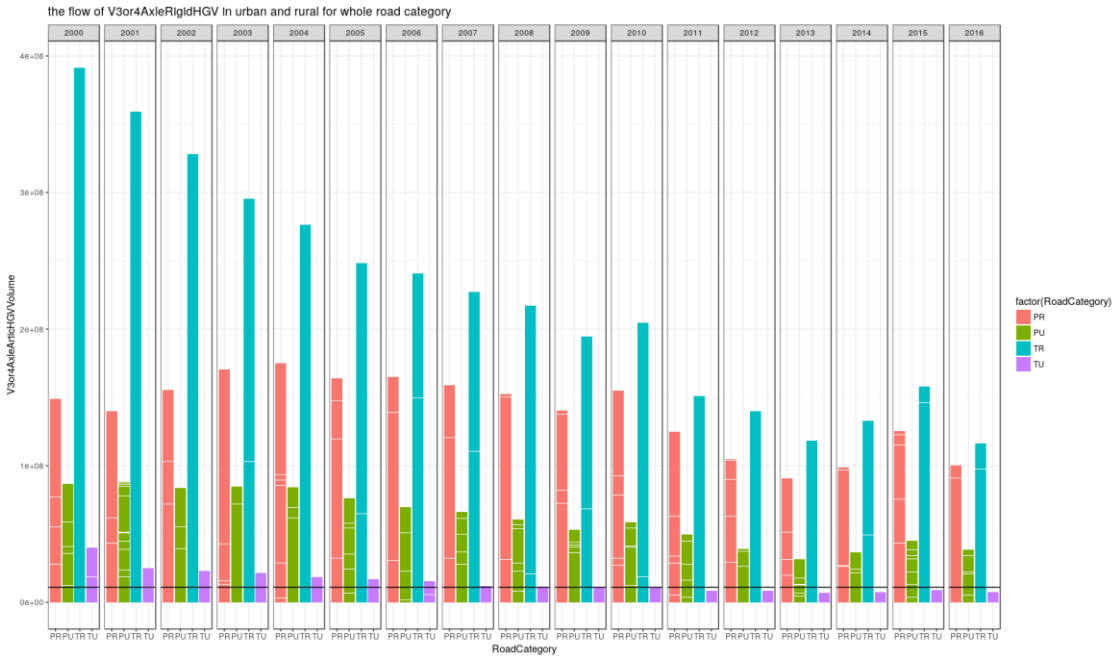


FIGURE7.11: THE FLOW OF V3OR4AXLERIGIDHGV IN URBAN AND RURAL FOR WHOLE ROAD CATEGORY

Pass the global average but decrease over years.

In TR road category the large usage.

7.2 .1 .9.9 V4or5AxleRigidHGV

```
g <- ggplot(UrbanRural,aes(RoadCategory,V4or5AxleRigidHGVVolume,fill =factor( RoadCategory)))
g+geom_bar(stat ="identity")+facet_grid(.~AADFYear)+theme_bw()+geom_hline(yintercept =
global_average )+labs(title = "the flow of V4or5AxleRigidHGV in urban and rural for whole road category")
```

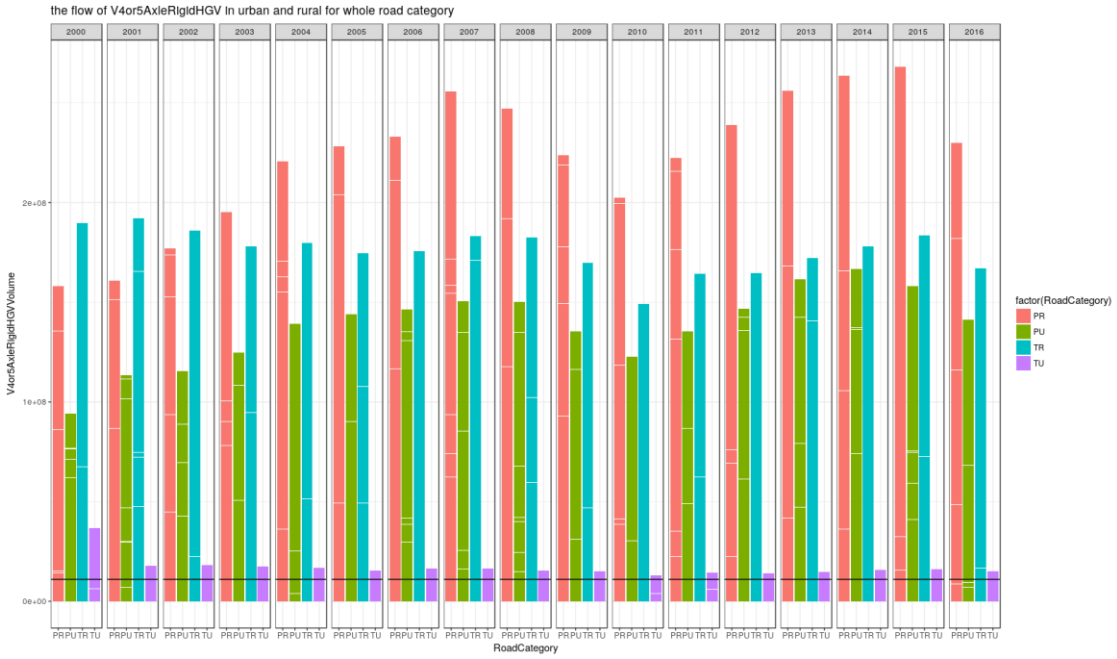


FIGURE7.12: THE FLOW OF V4OR5AXLERIGIDHGV IN URBAN AND RURAL FOR WHOLE ROAD CATEGORY

Pass the global average.

Still stand almost over years in all road categories except PR increase over year.

7.2 .1 .9.10 V5AxleArticHGV

```
g <- ggplot(UrbanRural,aes(RoadCategory,V5AxleArticHGVVolume,fill =factor( RoadCategory)))
g+geom_bar(stat ="identity")+facet_grid(.~AADFYear)+theme_bw()+geom_hline(yintercept =
global_average )+labs(title = "the flow of V5AxleRigidHGV in urban and rural for whole road category")
```

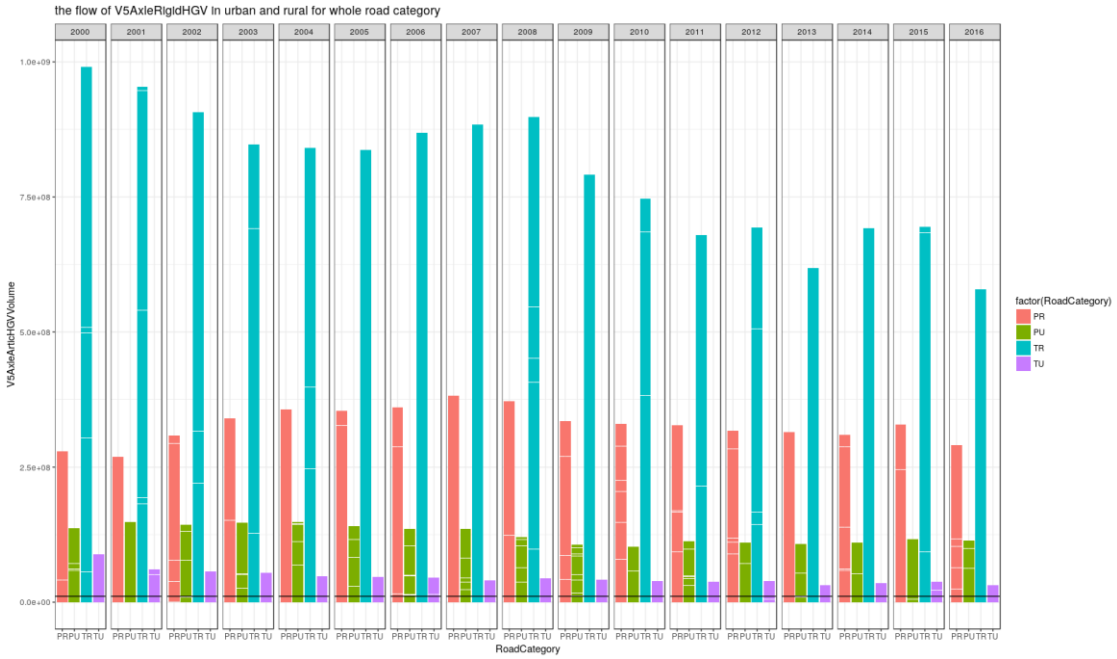


FIGURE7 13: THE FLOW OF V5AXLERIGIDHGV IN URBAN AND RURAL FOR WHOLE ROAD CATEGORY

Still stand almost in PR / PU / TU road category over years.

Useage is very large in TR road category and almost decrease over years.

7.2 .1 .9.11 V6orMoreAxleArticHGV

```
g <- ggplot(UrbanRural,aes(RoadCategory,V6orMoreAxleArticHGVVolume,fill =factor( RoadCategory)))
g+geom_bar(stat ="identity")+facet_grid(.~AADFYear)+theme_bw()+geom_hline(yintercept =
global_average )+labs(title = "the flow of V6ormoreAxleRigidHGV in urban and rural for whole road
category")
```

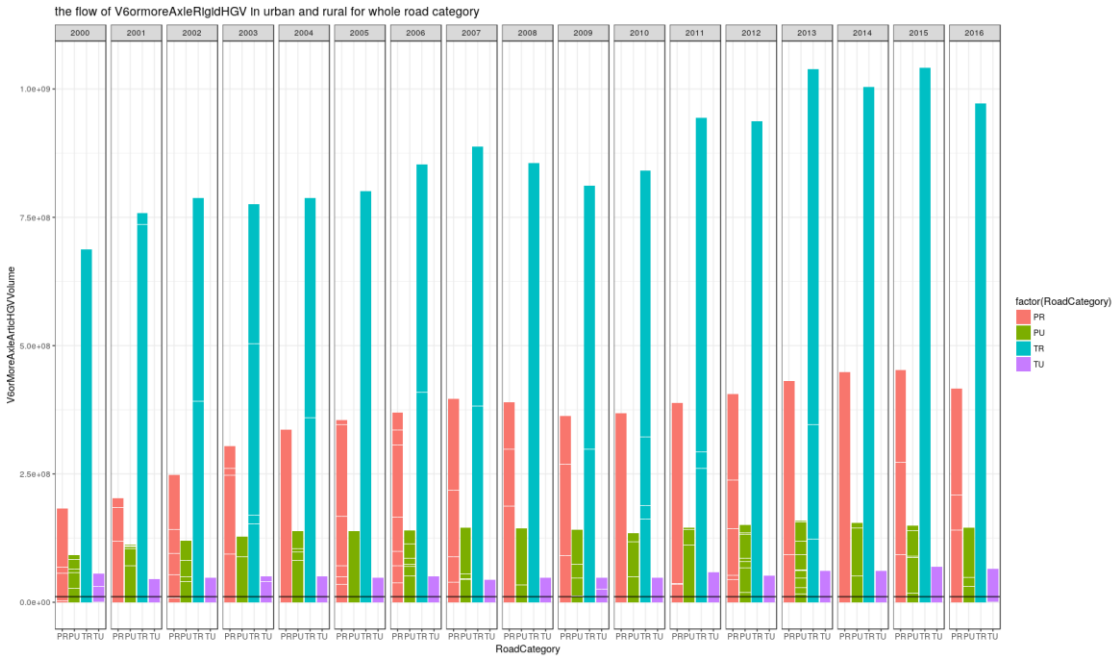


FIGURE7 14: THE FLOW OF V6ORMOREAXLERIGIDHGV IN URBAN AND RURAL FOR WHOLE ROAD CATEGORY

PR and TR road category increase over years. Other road category almost still stands.

The large usage in TR roads.

7.2 .1 .10 Make a report for each road pass the global average in urban and rural roads

```
UrbanRural <- TrafficAADF %>%
  select(AADFYear,Road,RoadCategory,PedalCyclesVolume,
         MotorcyclesVolume,BusesCoachesVolume,
         LightGoodsVehiclesVolume,
         V2AxleRigidHGVVolume,V3AxleRigidHGVVolume,
         CarsTaxisVolume,V3or4AxleArticHGVVolume,
         V4or5AxleRigidHGVVolume,V5AxleArticHGVVolume,
         V6orMoreAxleArticHGVVolume
         )
```

View of data

```
head(UrbanRural)
```

```
## # A tibble: 6 x 14
##   AADFYear Road  RoadCategory PedalCyclesVolume MotorcyclesVolume
##      <int> <chr> <chr>                   <dbl>             <dbl>
## 1     2000 M1    TM                      0.              144715.
## 2     2000 M621  TM                      0.               84417.
## 3     2000 M18   TM                      0.               36865.
## 4     2000 M62   TM                      0.              176988.
## 5     2000 M62   TM                      0.              254098.
## 6     2000 M62   TM                      0.              162695.
## # ... with 9 more variables: BusesCoachesVolume <dbl>,
## #   LightGoodsVehiclesVolume <dbl>, V2AxleRigidHGVVolume <dbl>,
## #   V3AxleRigidHGVVolume <dbl>, CarsTaxisVolume <dbl>,
## #   V3or4AxleArticHGVVolume <dbl>, V4or5AxleRigidHGVVolume <dbl>,
## #   V5AxleArticHGVVolume <dbl>, V6orMoreAxleArticHGVVolume <dbl>
```

7.2 .1 .11 Group data with needed columns

```
UrbanRuralRoads <- group_by(UrbanRural,AADFYear,RoadCategory,Road)
```

7.2 .1 .12 Gather the data

```
gather_vehicles <- gather(UrbanRuralRoads,vehicles_type,count,-c(AADFYear,Road,RoadCategory))
```

View of data

```
head(gather_vehicles)
```

```
## # A tibble: 6 x 5
## # Groups:   AADFYear, RoadCategory, Road [4]
##   AADFYear Road  RoadCategory vehicles_type     count
##      <int> <chr> <chr>       <chr>            <dbl>
## 1     2000 M1    TM          PedalCyclesVolume    0.
## 2     2000 M621  TM          PedalCyclesVolume    0.
## 3     2000 M18   TM          PedalCyclesVolume    0.
## 4     2000 M62   TM          PedalCyclesVolume    0.
## 5     2000 M62   TM          PedalCyclesVolume    0.
## 6     2000 M62   TM          PedalCyclesVolume    0.
```

7.2 .1 .13 Filter data which count of vehicles over come the global average

```
UrbanRuralRoads <- filter(gather_vehicles,count >= global_average)
```

This report for all roads and its category over years which overcome the global average.

```
head(UrbanRuralRoads)
```

```
## # A tibble: 6 x 5
## # Groups:   AADFYear, RoadCategory, Road [3]
##   AADFYear Road  RoadCategory vehicles_type            count
##      <int> <chr> <chr>        <chr>                   <dbl>
## 1     2000 M1    TM           LightGoodsVehiclesVolume 12016530.
## 2     2000 M62   TM           LightGoodsVehiclesVolume 15304059.
## 3     2000 M62   TM           LightGoodsVehiclesVolume 13113253.
## 4     2000 A1    TM           LightGoodsVehiclesVolume 13313448.
## 5     2000 M1    TM           LightGoodsVehiclesVolume 12384552.
## 6     2000 M1    TM           LightGoodsVehiclesVolume 15601268.
```

7.2 .1 .14 Divide data into two parts Rural and Urban, then see if the flow in the roads which pass global average is increase or less overyears.

7.2 .1 .14.1 first, start with Rural roads

```
RuralRoads <- filter(UrbanRuralRoads,RoadCategory=="TR"|RoadCategory=="PR") %>%
group_by(AADFYear,vehicles_type)
```

View of data

```
head(RuralRoads)
```

```
## # A tibble: 6 x 5
## # Groups:   AADFYear, vehicles_type [3]
##   AADFYear Road  RoadCategory vehicles_type            count
##      <int> <chr> <chr>        <chr>                   <dbl>
## 1     2000 A1    TR           LightGoodsVehiclesVolume 14170694.
## 2     2000 A1    TR           LightGoodsVehiclesVolume 13546847.
## 3     2001 A1    TR           LightGoodsVehiclesVolume 11279376.
## 4     2001 A1    TR           LightGoodsVehiclesVolume 14516488.
## 5     2001 A1    TR           LightGoodsVehiclesVolume 12714888.
## 6     2002 A1    TR           LightGoodsVehiclesVolume 11568456.
```

The average mean of each year

```
average_year <- summarise(RuralRoads,average =mean(count))
head(average_year)
```

```
## # A tibble: 6 x 3
## # Groups:   AADFYear [3]
##   AADFYear vehicles_type            average
##      <int> <chr>                    <dbl>
## 1     2000 CarsTaxisVolume          22679233.
## 2     2000 LightGoodsVehiclesVolume 13294658.
## 3     2001 CarsTaxisVolume          23032730.
## 4     2001 LightGoodsVehiclesVolume 13488716.
## 5     2002 CarsTaxisVolume          23431946.
## 6     2002 LightGoodsVehiclesVolume 13420237.
```

```
g <- ggplot(average_year,aes(AADFYear,average))
g+geom_point()+geom_smooth(method = "lm",se = F)+geom_smooth(method = "loess",color ="red",se = F)+theme_bw()+facet_grid(.~vehicles_type)+labs(title = "the average of each vechiles which pass the global average over year in Rural")
```

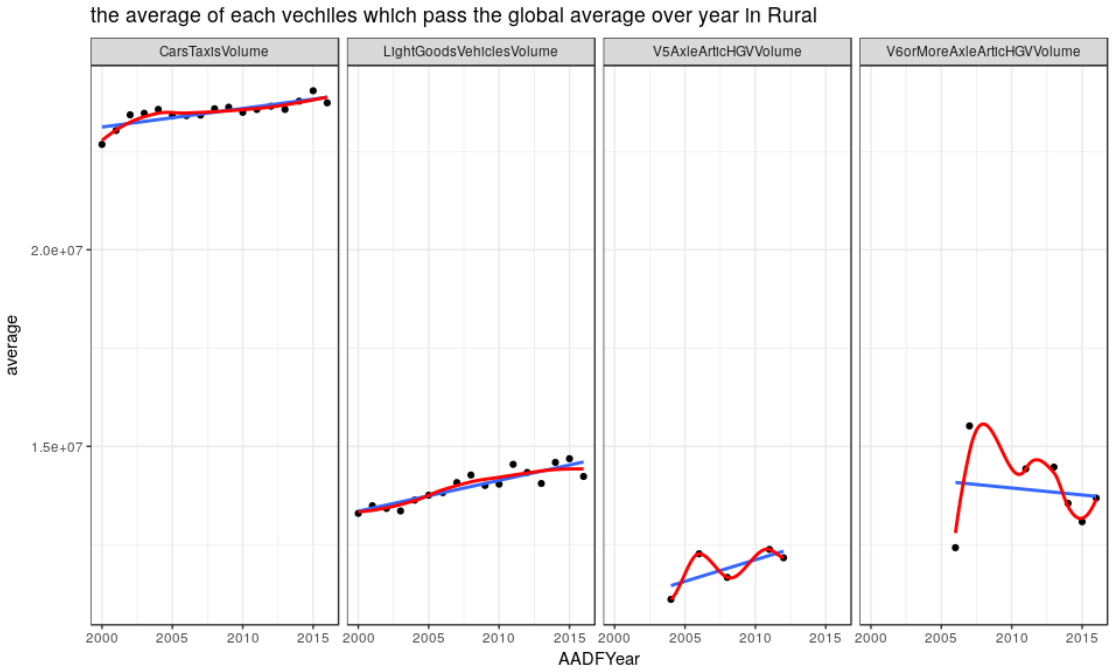


FIGURE7 15:THE AVERAGE OF EACH VECHILES WHICH PASS THE GLOBAL AVERAGE OVER YEAR IN RURAL

We see that only four types of cars pass the global average, taxis, light goods, V5, V6.

See that the average of cars is increasng over year except the V6 is decreasing.

7.2 .1 .14.2 Second, with Urban roads

```
UrbanRoads <- filter(UrbanRuralRoads,RoadCategory=="TU"|RoadCategory=="PU") %>%
  group_by(AADFYear,vehicles_type)
```

View of data

```
head(UrbanRoads)
```

```
## # A tibble: 6 x 5
## # Groups:   AADFYear, vehicles_type [6]
##   AADFYear Road  RoadCategory vehicles_type            count
##      <int> <chr> <chr>        <chr>                   <dbl>
## 1     2014 A8    TU           LightGoodsVehiclesVolume 11853003.
## 2     2015 A8    TU           LightGoodsVehiclesVolume 12467779.
## 3     2016 A8    TU           LightGoodsVehiclesVolume 13396128.
## 4     2000 A40   TU           LightGoodsVehiclesVolume 14063019.
```

```
## 5    2001 A12   PU        LightGoodsVehiclesVolume 11516407.
## 6    2003 A40   PU        LightGoodsVehiclesVolume 16268619.
```

The average in each year

```
average_year <- summarise(UrbanRoads,average = mean(count))
```

View of data

```
head(average_year)
```

```
## # A tibble: 6 x 3
## # Groups:   AADFYear [4]
##   AADFYear vehicles_type           average
##      <int> <chr>                     <dbl>
## 1    2000 CarsTaxisVolume         17379893.
## 2    2000 LightGoodsVehiclesVolume 14063019.
## 3    2001 CarsTaxisVolume         17505630.
## 4    2001 LightGoodsVehiclesVolume 11516407.
## 5    2002 CarsTaxisVolume         17463086.
## 6    2003 CarsTaxisVolume         17596399.
```

```
g <- ggplot(average_year,aes(AADFYear,average))
g+geom_point()+geom_smooth(method = "lm",se = F)+theme_bw()+facet_grid(.~vehicles_type)+labs(title = "the average of each vechiles which pass the global average over year in Rural")
```

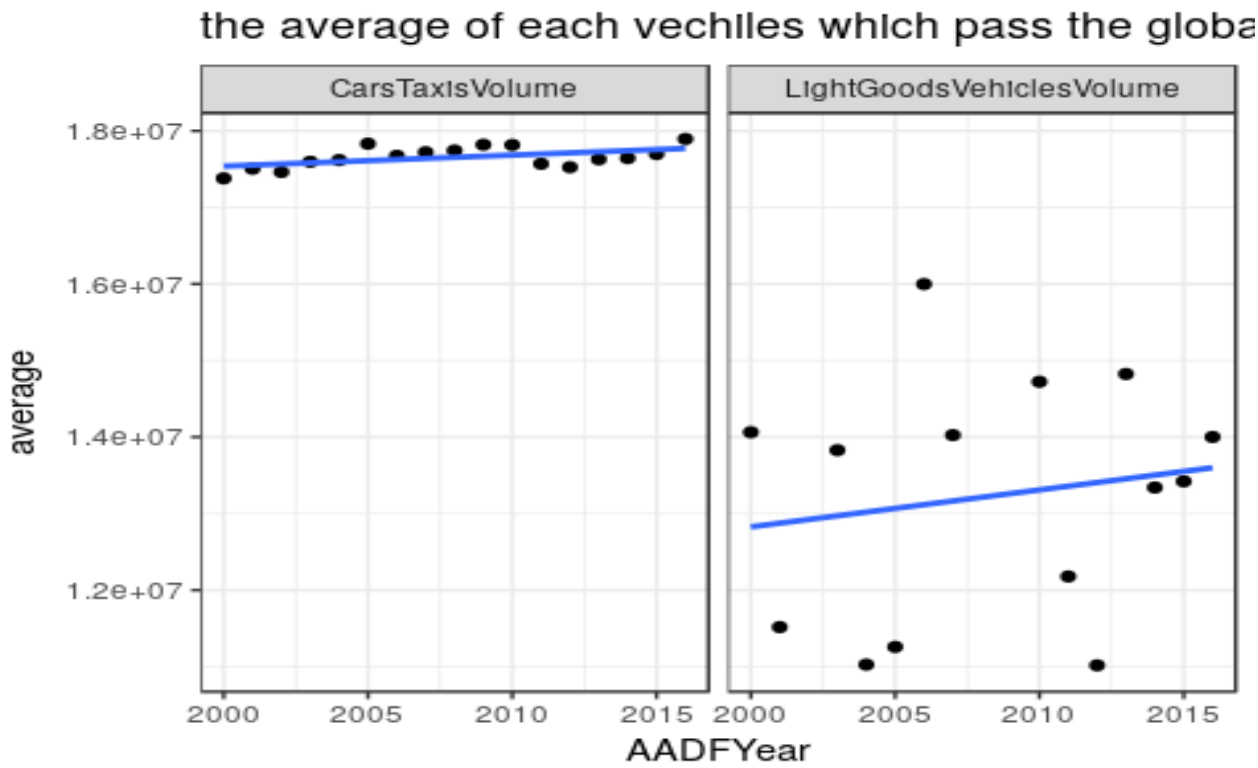


FIGURE7 16: THE AVERAGE OF EACH VECHILES WHICH PASS THE GLOBAL AVERAGE OVER YEAR IN RURAL

We see that only two types of cars pass the global average taxis and light goods.

We see that the average of taxis still stand almost and light goods increase over year.

7.2 .1 .15 Go to see the average of vehicles in all roads over years.

```
gather_vehicles <- group_by(gather_vehicles,AADFYear,vehicles_type,RoadCategory)
average_year <- summarise(gather_vehicles,average =mean(count))
g <- ggplot(average_year,aes(AADFYear,average))
g+geom_point()+geom_smooth(method = "lm",se =
F)+theme_bw()+facet_grid(RoadCategory~vehicles_type)+labs(title = "the average of vehicles in all roads over years")
```

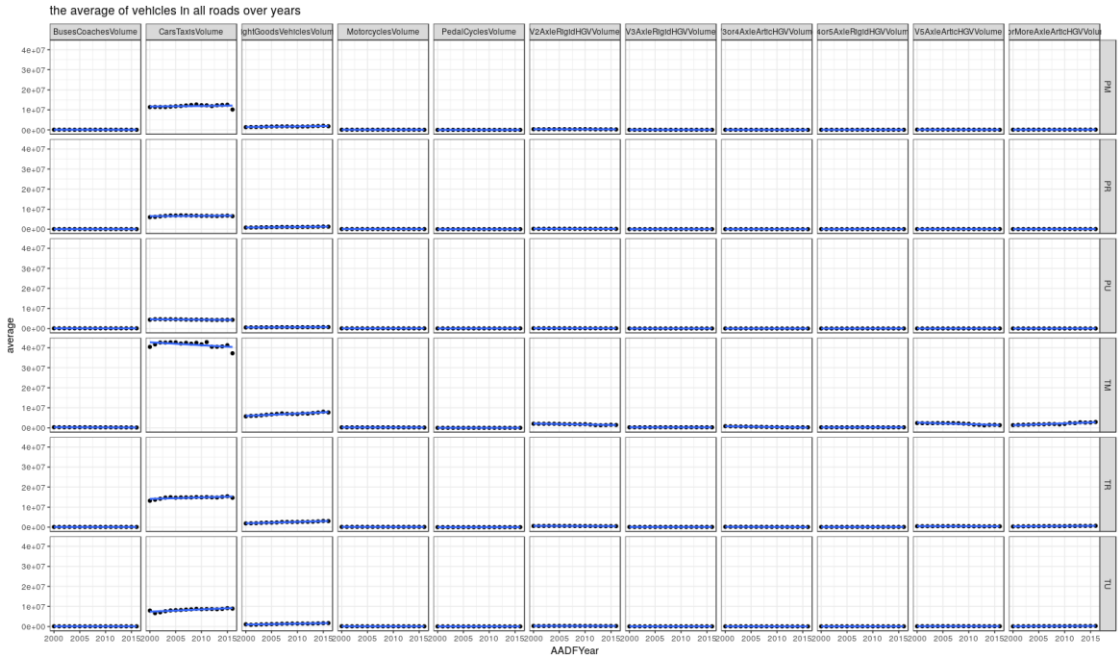


FIGURE7 17: THE AVERAGE OF VEHICLES IN ALL ROADS OVER YEARS

We recoginze that taxis increase over years in all road categery

7.2 .2 Second problem

Problem: How has changing traffic flow impacted accidents?

Loading libraries

```
library(dplyr)
library(ggplot2)
library(tidyr)
```

7.2 .2 .1 Prepare data

We need select the columns, work on traffic data first to select the columns which we worked on.

We filter the selected data by years to get the same years in traffic and accident files

We remove the 2008 and all years that not have accidents records.

We remove rows which have missing value.

```
TrafficFirstGoal <- TrafficAADF %>%
 select(AADFYear,CP,Region,PedalCyclesVolume,
       MotorcyclesVolume,BusesCoachesVolume,
       LightGoodsVehiclesVolume,
       V2AxleRigidHGVVolume,V3AxleRigidHGVVolume,
       CarsTaxisVolume,V3or4AxleArticHGVVolume,
       V4or5AxleRigidHGVVolume,
       V5AxleArticHGVVolume,
       V6orMoreAxleArticHGVVolume,
       allVolume,allVehicles
 ) %>%

 filter(AADFYear != 2000,AADFYear !=2001,AADFYear !=2002,AADFYear !=2003,
       AADFYear !=2004,AADFYear !=2008,AADFYear !=2015,AADFYear !=2016)%>%
 group_by(AADFYear)
```

7.2 .2 .2 Selected needed columns in second data

```
accFirstGoal <-AllAccidents %>%
 select(Accident_Index,Number_of_Vehicles,Date,Year) %>%
 group_by(Year,Date)
```

7.2 .2 .3 Check if have missing value

```
sum(is.na(accFirstGoal))
```

```
## [1] 0
```

```
sum(is.na(TrafficFirstGoal))
```

```
## [1] 0
```

Not have missing values.

7.2 .2 .4 Visualize data

To know how chaging traffic flow over years impact on accidents by number of vehicles and traffic volume

Let's see the change of average of traffic volume over years

```
average_year <- summarise(TrafficFirstGoal,meanVolume = mean(allVolume))
head(average_year)
```

```
## # A tibble: 6 x 2
##   AADFYear meanVolume
##      <int>      <dbl>
## 1     2005  11064777.
## 2     2006  11214578.
## 3     2007  11210355.
## 4     2009  11063780.
## 5     2010  10935574.
## 6     2011  11027359.
```

```
g <- ggplot(average_year,aes(AADFYear,meanVolume))
g+geom_point()+geom_smooth(method = "lm",se = FALSE)+theme_bw()+geom_smooth(method = "loess",color="red",se = FALSE)+scale_x_discrete(limits=(2005:2014))+labs(title = "average of traffic volume over years")
```

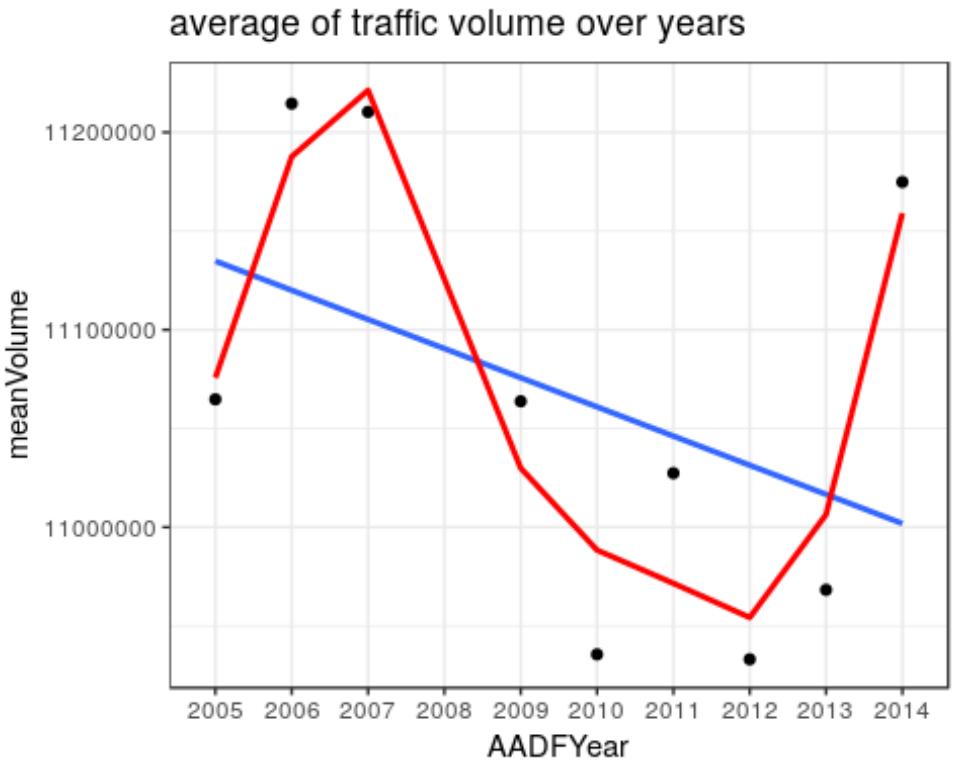


FIGURE7 18: AVERAGE OF TRAFFIC VOLUME OVER YEARS

We see that the average of the traffic volume less over year.

7.2 .2 .5 Impact of average of allvehicles on each road over years

```
average_year <- summarize(TrafficFirstGoal,meanVehicle = mean(allVehicles))
head(average_year)
```

```
## # A tibble: 6 x 2
##   AADFYear meanVehicle
##      <int>       <dbl>
## 1     2005      21414.
## 2     2006      21712.
## 3     2007      21601.
## 4     2009      21298.
## 5     2010      21090.
## 6     2011      21150.
```

```
g <- ggplot(average_year,aes(AADFYear,meanVehicle))
g+geom_point()+geom_smooth(method = "lm",se = FALSE)+geom_smooth(method = "loess",se =
FALSE,color = "red")+theme_bw()+geom_smooth(method = "loess",color="red",se =
FALSE)+scale_x_discrete(limits=(2005:2014))+labs(title ="average of allvehicles on each road over years")
```

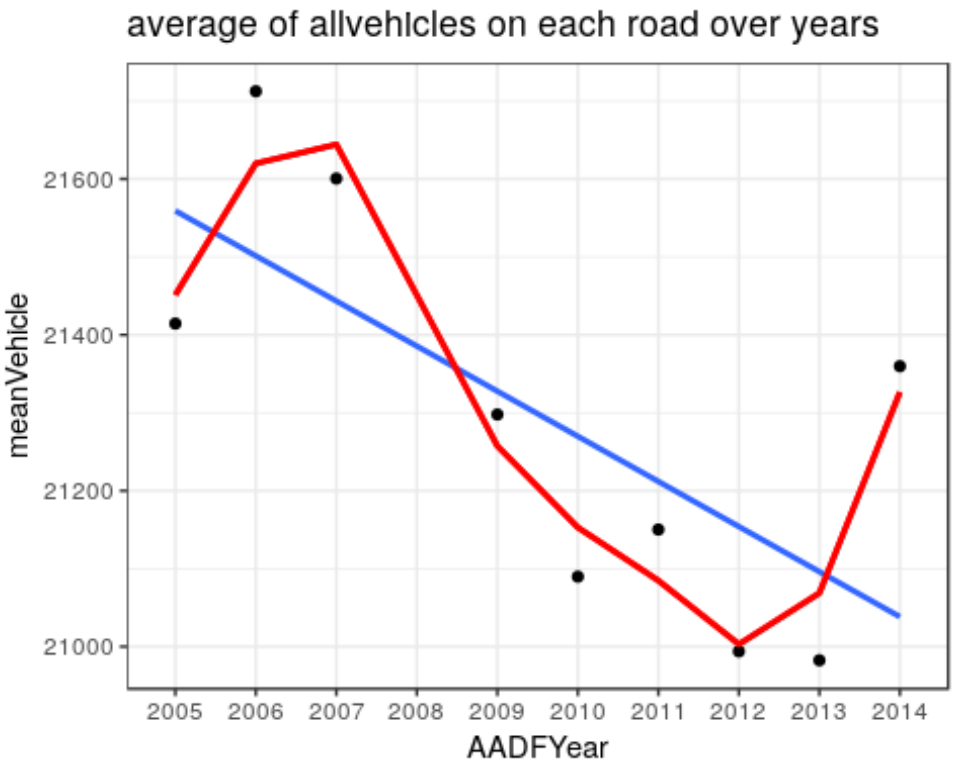


FIGURE7 19: AVERAGE OF ALLVEHICLES ON EACH ROAD OVER YEARS

We see that over year's traffic less many than year before, so we look at the accidents average over year too.

7.2 .2 .6 Convert to annual average

To make the best and specified result we sould take care that the traffic flow be the average daily in year so we must convert to annual average. Same as get the all accident on a day and then calculate the annual average.

7.2 .2 .6.1 Get the sum of accidents be done on every day

```
average_day<- summarize(accFirstGoal,number_accident_day=n())
head(average_day)
```

```
## # A tibble: 6 x 3
## # Groups:   Year [1]
##    Year Date       number_accident_day
##   <int> <chr>                    <int>
## 1  2005 01/01/2005                 308
## 2  2005 01/02/2005                 628
## 3  2005 01/03/2005                 590
## 4  2005 01/04/2005                 464
## 5  2005 01/05/2005                 486
## 6  2005 01/06/2005                 655
```

7.2 .2 .6.2 Group the data result

Group the data result by year to can get the mean of year of accidents.

We can get the annually average of accidents

```
average_year<-group_by(average_day,Year)
average_year <- summarize(average_year,average_in_year = mean(number_accident_day))
head(average_year)
```

```
## # A tibble: 6 x 2
##    Year average_in_year
##   <int>          <dbl>
## 1  2005          544.
## 2  2006          518.
## 3  2007          499.
## 4  2009          448.
## 5  2010          423.
## 6  2011          415.
```

```
g<- ggplot(average_year,aes(Year,average_in_year))
g+geom_point()+geom_smooth(method = "lm",se = FALSE)+geom_smooth(method = "loess",se = FALSE,color = "red")+theme_bw()+scale_x_discrete(limits=(2005:2014))+labs(title = "daily annually average of accidents")
```

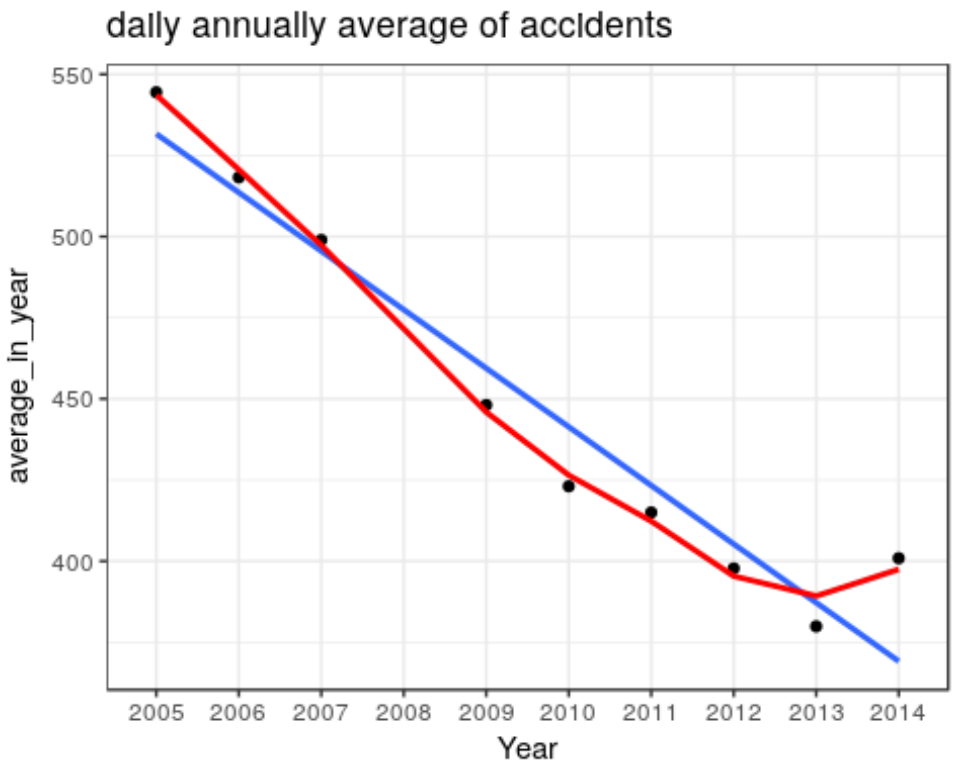


FIGURE7 20: DAILY ANNUALLY AVERAGE OF ACCIDENTS

7.2 .2 .6.3 Get the sum of number of vehicles accidents be done on every day

```
average_day<- summarize(accFirstGoal,sum_accident_day=sum(Number_of_Vehicles))
```

Group the data result by year to can get the mean of year of number of vehicles of accidents .

```
average_year<-group_by(average_day,Year)
```

We can get the annual daily average of number of vehicles of accidents

```
average_year <- summarize(average_year,number_of_vehicles = mean(sum_accident_day))
head(average_year)
```

```
## # A tibble: 6 x 2
##    Year number_of_vehicles
##   <int>              <dbl>
## 1  2005              1003.
## 2  2006               954.
## 3  2007               918.
## 4  2009               818.
## 5  2010               771.
## 6  2011               757.
```

7.2 .2 .6.4 We visualize the result to see the average of Number_of_Vehicles of accidents over years

```
g<- ggplot(average_year,aes(Year,number_of_vehicles))
g+geom_point()+geom_smooth(method = "loess",se = F)+geom_smooth(method = "loess",se = FALSE,color = "red")+geom_smooth(method = "lm",se = F)+theme_bw()+
scale_x_discrete(limits=(2005:2014))+labs(title = "average of Number_of_Vehicles of accidents over years")
```

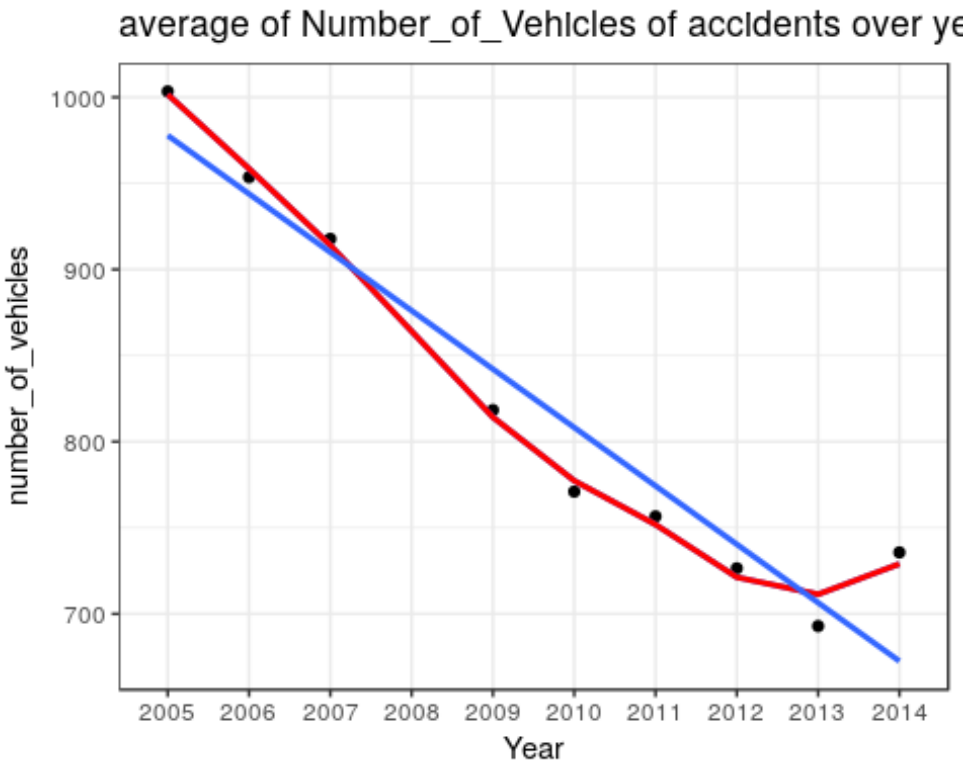


FIGURE7 21: AVERAGE OF NUMBER_OF_VEHICLES OF ACCIDENTS OVER YEARS

We see that the accidents less over year.

7.2 .2 .7 We see the most year accidents be done

```
g<-ggplot(accFirstGoal,aes(Year,fill = factor(Year)))
g+geom_bar()+scale_x_discrete(limits=(2005:2014))+labs(title = "accident rate over years")
```

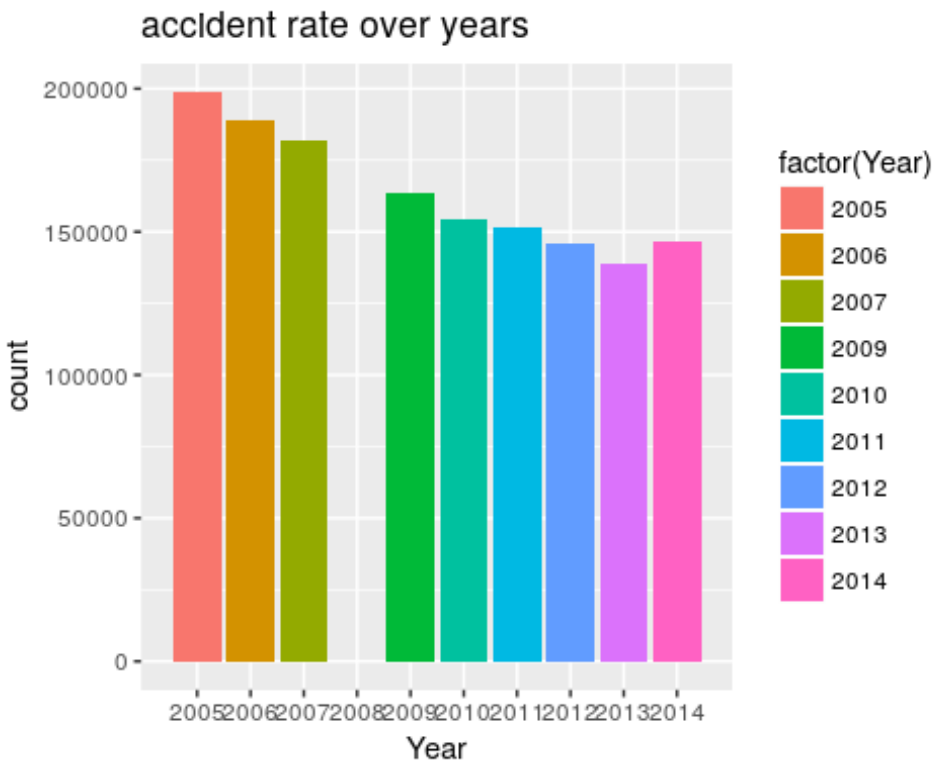


FIGURE7 22: ACCIDENT RATE OVER YEARS

We see how accident rates less over time.

When look to the two graph we see when the traffic flow less over year the accidents less too.

7.2 .2 .8 we look to the number of all kind of cars and how changed over years

7.2 .2 .8.1 Pedal cycles

```
pedalCycle<- summarise(TrafficFirstGoal,average_cycle = mean(PedalCyclesVolume))
g <- ggplot(pedalCycle,aes(AADFYear,average_cycle))
g+geom_point()+geom_smooth(method = "lm",se = F)+geom_smooth(method =
"loess",se=F,color="red")+theme_bw()+scale_x_discrete(limits=(2005:2014))+labs(title = "pedal cycle
average over year")
```

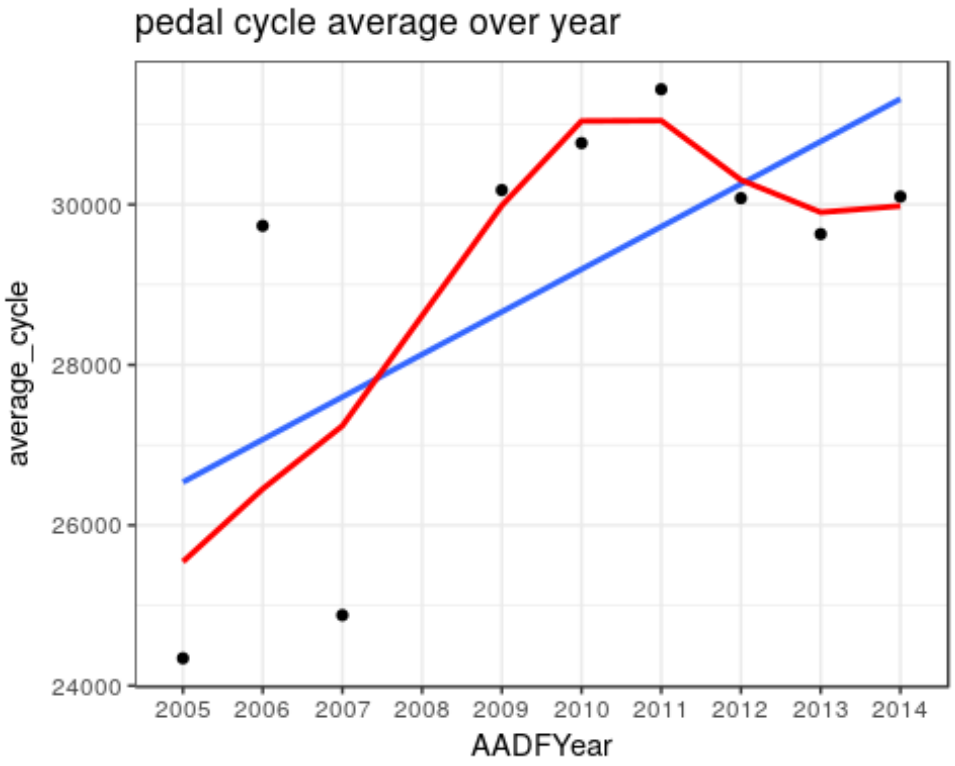


FIGURE7 23: PEDAL CYCLE AVERAGE OVER YEAR

We see that the use of cycles increase over years

7.2 .2 .8.2 Motor cycles vehicles

```
motorcycle<- summarise(TrafficFirstGoal,average_motor = mean(MotorcyclesVolume))
g <- ggplot(motorcycle,aes(AADFYear,average_motor))
g+geom_point()+geom_smooth(method = "lm",se = F)+geom_smooth(method =
"loess",se=F,color="red")+theme_bw()+scale_x_discrete(limits=(2005:2014))+labs(title = "motor cycle
average over year")
```

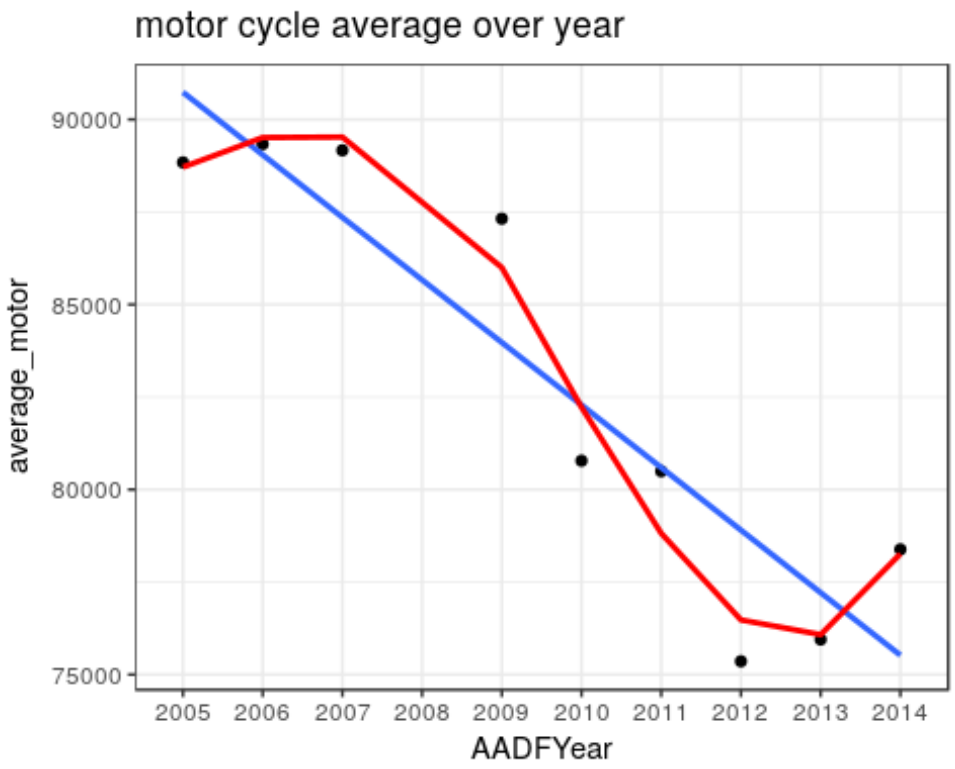


FIGURE7 24: MOTOR CYCLE AVERAGE OVER YEAR

We see that the motor cycle less over time.

7.2 .2 .8.3 Buses coaches vehicles

```
busescoaches<- summarise(TrafficFirstGoal,average_buses = mean(BusesCoachesVolume))
g <- ggplot(busescoaches,aes(AADFYear,average_buses))
g+geom_point()+geom_smooth(method = "lm",se = F)+geom_smooth(method =
"loess",se=F,color="red")+theme_bw()+scale_x_discrete(limits=(2005:2014))+labs(title = "buses coaches
average over year")
```

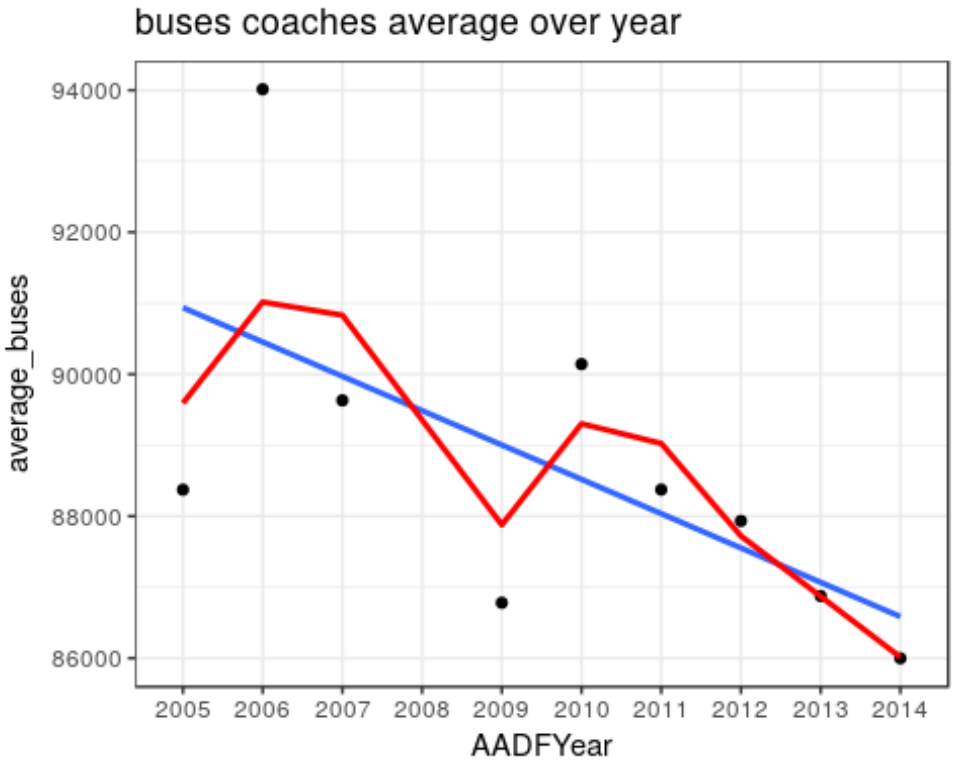


FIGURE7 25: BUSES COACHES AVERAGE OVER YEAR

Buses less slowly as fell it not change.

7.2 .2 .8.4 Car taxis vehicles

```
taxis<- summarise(TrafficFirstGoal,average_taxis = mean(CarsTaxisVolume))
g <- ggplot(taxis,aes(AADFYear,average_taxis))
g+geom_point()+geom_smooth(method = "lm",se = F)+geom_smooth(method =
"loess",se=F,color="red")+theme_bw()+scale_x_discrete(limits=(2005:2014))+labs(title = "taxis average
over year")
```

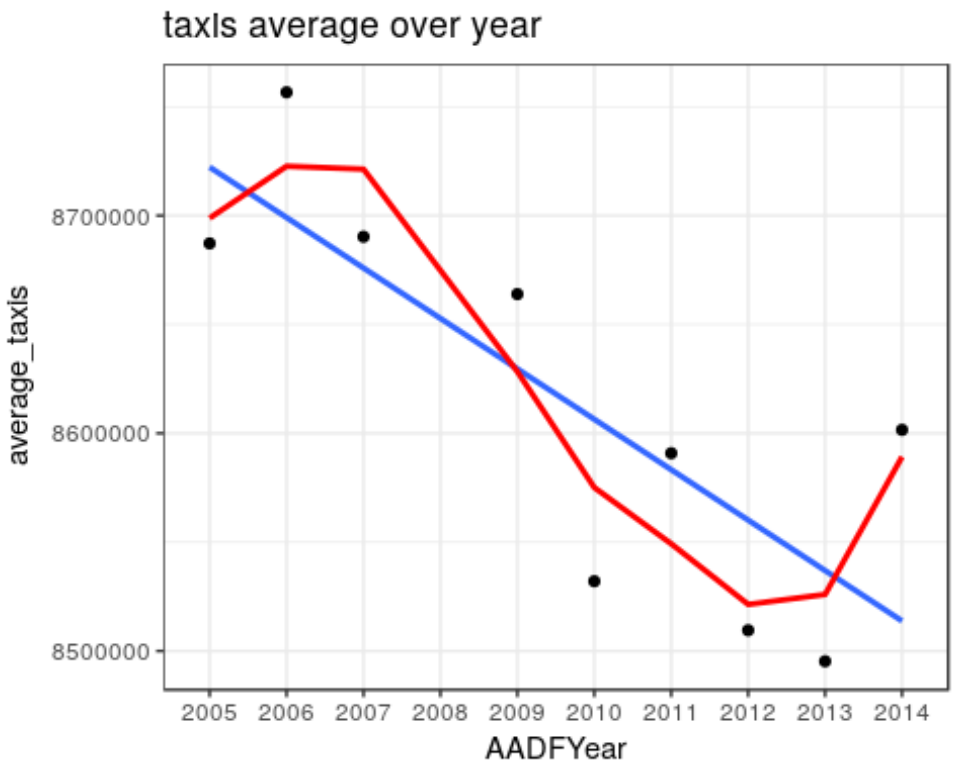


FIGURE7 26: TAXIS AVERAGE OVER YEAR

Taxis less over years.

7.2 .2 .8.5 Light goods vehicles

```
lightGood<- summarise(TrafficFirstGoal,average_light = mean(LightGoodsVehiclesVolume))
g <- ggplot(lightGood,aes(AADFYear,average_light))
g+geom_point()+geom_smooth(method = "lm",se = F)+geom_smooth(method =
"loess",se=F,color="red")+theme_bw()+scale_x_discrete(limits=(2005:2014))+labs(title = "light goods
average over year")
```

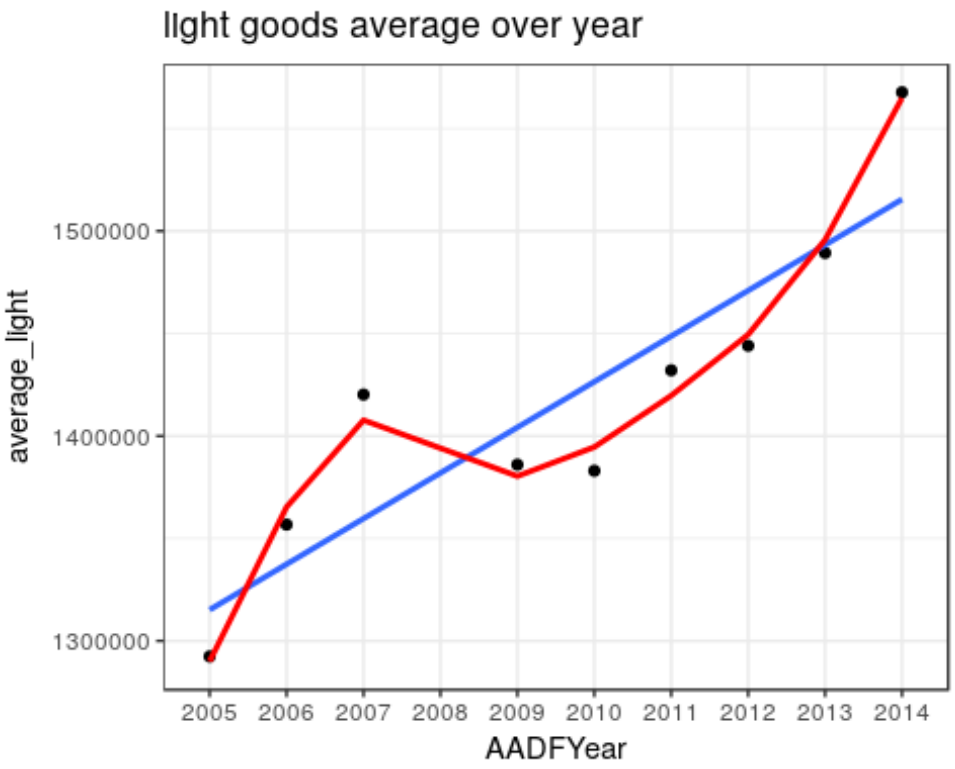


FIGURE7 27: LIGHT GOODS AVERAGE OVER YEAR

We see the light good vechicles increased over years.

7.2 .2 .8.6 hgv2 vehicles

```
hgv<- summarise(TrafficFirstGoal,average_v2 = mean(V2AxleRigidHGVVolume))
g <- ggplot(hgv,aes(AADFYear,average_v2))
g+geom_point()+geom_smooth(method = "lm",se = F)+geom_smooth(method =
"loess",se=F,color="red")+theme_bw()+scale_x_discrete(limits=(2005:2014))+labs(title = "V2AxleRigidHGV
average over year")
```

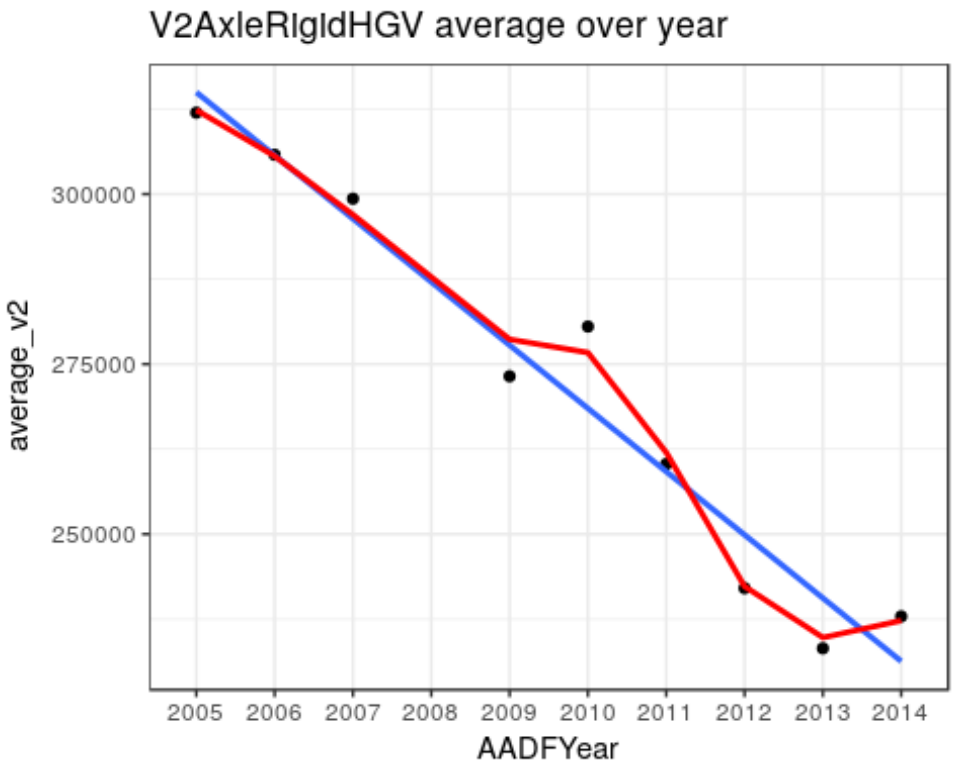


FIGURE7 28: V2AXLERIGIDHGV AVERAGE OVER YEAR

We see that V2AxleRigidHGV less over years.

7.2 .2 .8.7 hgv3 vehicles

```
hgv<- summarise(TrafficFirstGoal,average_v3 = mean(V3AxleRigidHGVVolume))
g <- ggplot(hgv,aes(AADFYear,average_v3))
g+geom_point()+geom_smooth(method = "lm",se = F)+geom_smooth(method =
"loess",se=F,color="red")+theme_bw()+scale_x_discrete(limits=(2005:2014))+labs(title = "V3AxleRigidHGV
average over year")
```

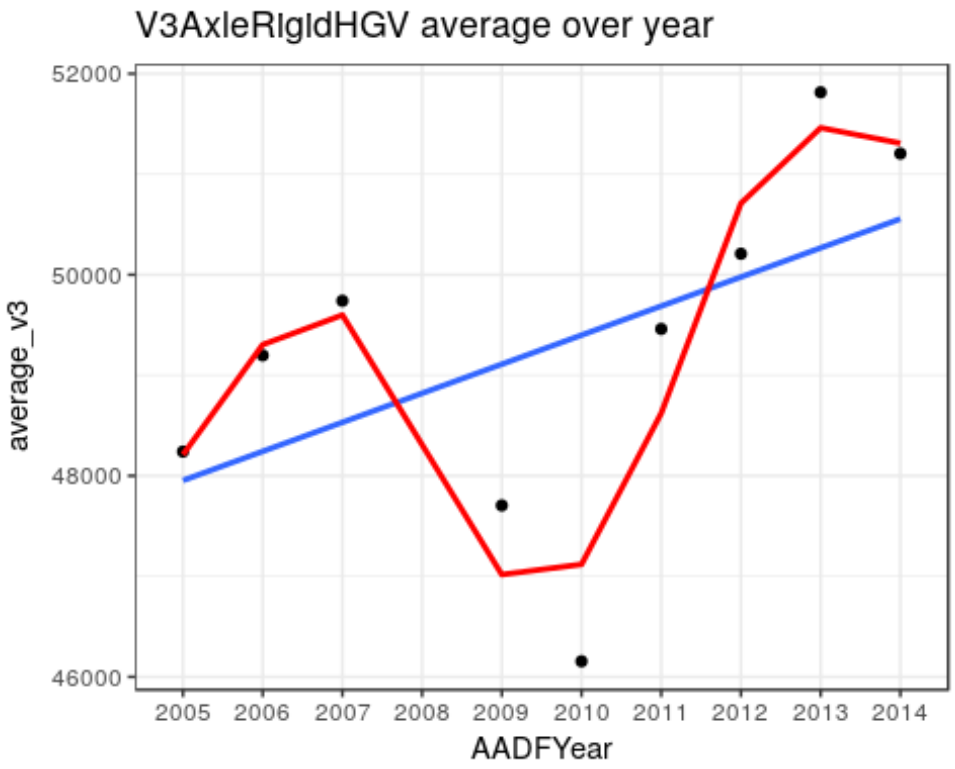


FIGURE7 29: V3AXLERIGIDHGV AVERAGE OVER YEAR

V3AxleRigidHGV still stand but increase slowly over years.

7.2 .2 .8.8 hgv34 vehicles

```
hgv<- summarise(TrafficFirstGoal,average_v34 = mean(V3or4AxleArticHGVVolume))
g <- ggplot(hgv,aes(AADFYear,average_v34))
g+geom_point()+geom_smooth(method = "lm",se = F)+geom_smooth(method =
"loess",se=F,color="red")+theme_bw()+scale_x_discrete(limits=(2005:2014))+labs(title =
"V3or4AxleRigidHGV average over year")
```

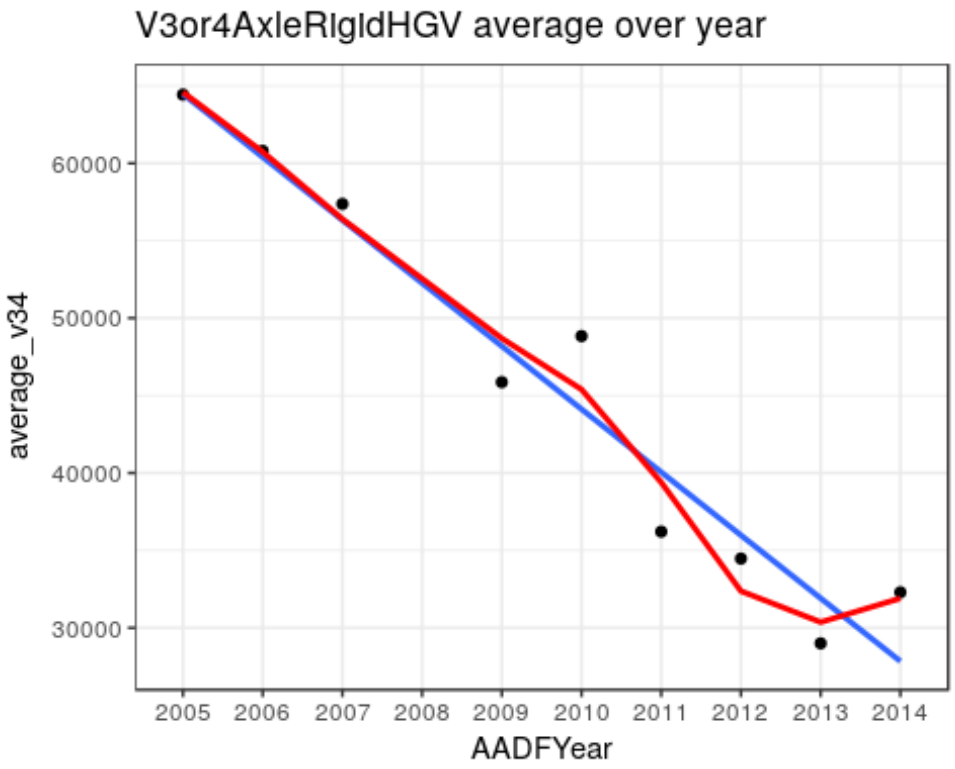


FIGURE7 30: V3OR4AXLERIGIDHGV AVERAGE OVER YEAR

We see this V3or4AxleRigidHGV still stand.

7.2 .2 .8.9 hgv5 vehicles

```
hgv<- summarise(TrafficFirstGoal,average_v5 = mean(V5AxleArticHGVVolume))
g <- ggplot(hgv,aes(AADFYear,average_v5))
g+geom_point()+geom_smooth(method = "lm",se = F)+geom_smooth(method =
"loess",se=F,color="red")+theme_bw()+scale_x_discrete(limits=(2005:2014))+labs(title = "V5AxleRigidHGV
average over year")
```

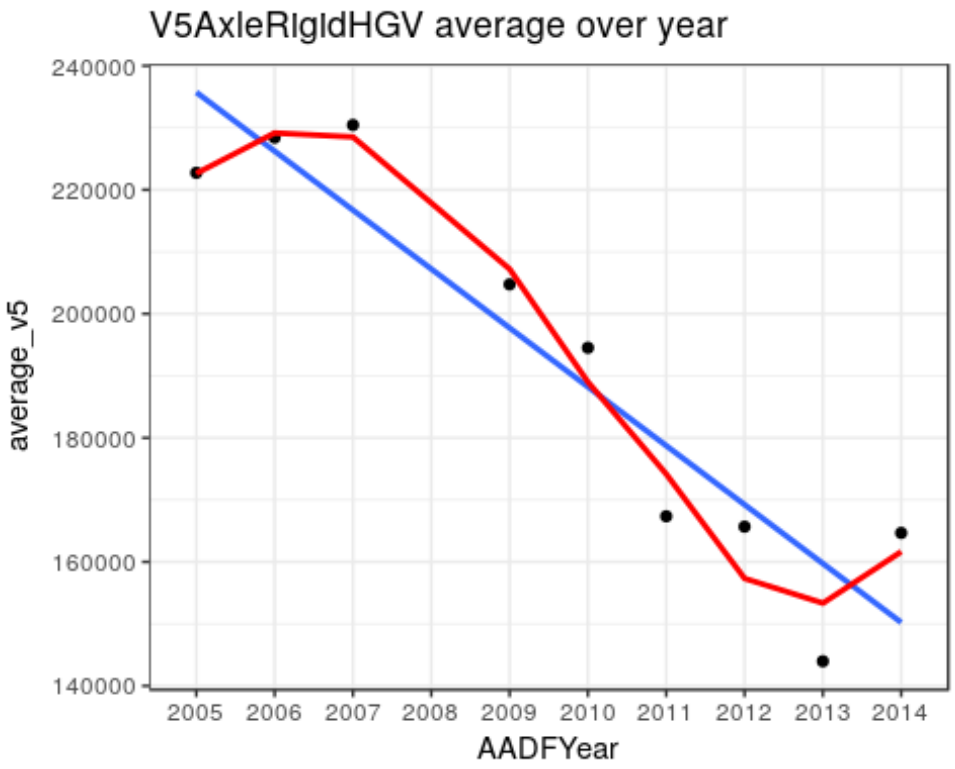


FIGURE7 31: V5AXLERIGIDHGV AVERAGE OVER YEAR

We see V5AxleRigidHGV less over years.

7.2 .2 .8.10 hgv45 vehicles

```
hgv<- summarise(TrafficFirstGoal,average_v45 = mean(V4or5AxleRigidHGVVolume))
g <- ggplot(hgv,aes(AADFYear,average_v45))
g+geom_point()+geom_smooth(method = "lm",se = F)+geom_smooth(method =
"loess",se=F,color="red")+theme_bw()+scale_x_discrete(limits=(2005:2014))+labs(title =
"V4or5AxleRigidHGV average over year")
```

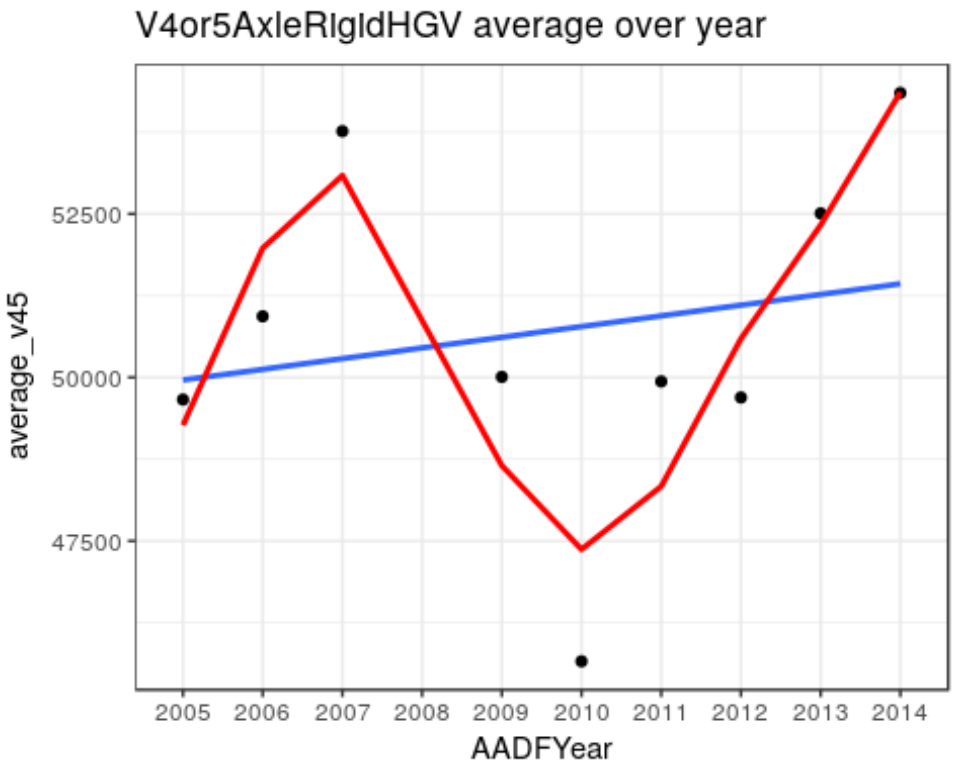


FIGURE7 32: V4OR5AXLERIGIDHGV AVERAGE OVER YEAR

We see V4or5AxleRigidHGV still stand.

7.2 .2 .8.11 hgv6

```
hgv<- summarise(TrafficFirstGoal,average_v6 = mean(V6orMoreAxleArticHGVVolume))
g <- ggplot(hgv,aes(AADFYear,average_v6))
g+geom_point()+geom_smooth(method = "lm",se = F)+geom_smooth(method =
"loess",se=F,color="red")+theme_bw()+scale_x_discrete(limits=(2005:2014))+labs(title =
"V6ormoreAxleRigidHGV average over year")
```

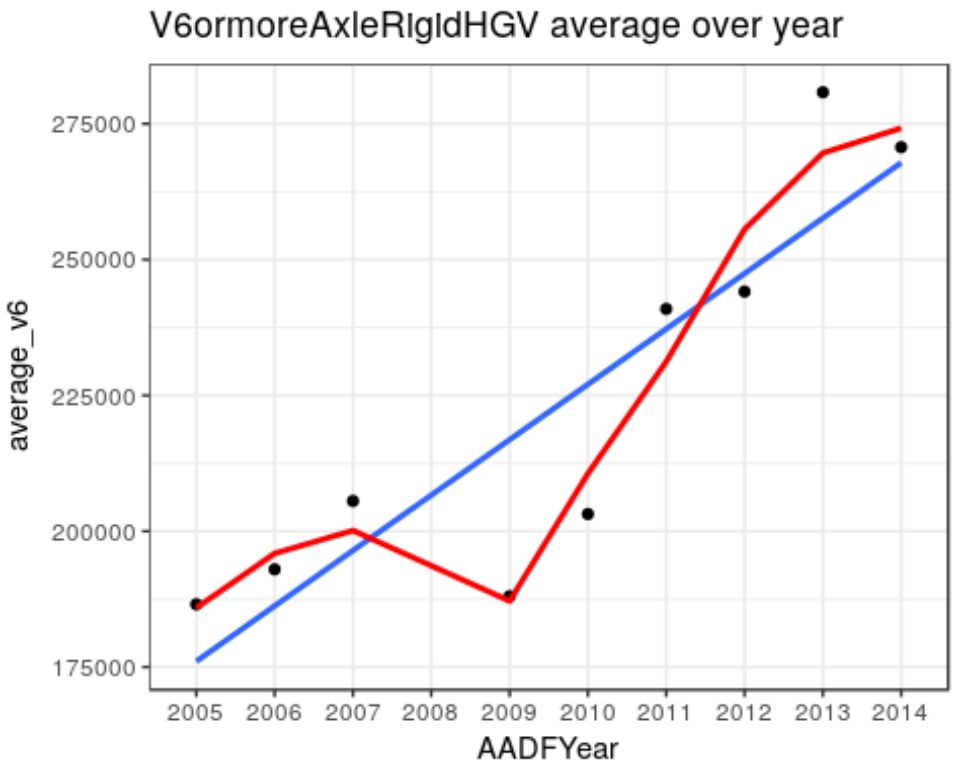


FIGURE7 33: V6ORMOREAXLERIGIDHGV AVERAGE OVER YEAR

We see that increase of V6ormoreAxleRigidHGV.

Note: to overcome the busy flow we must control the average of the following cars V6orMoreAxleArticHGV / LightGoodsVehicles / pedal cycles.

7.2 .2 .8.12 the difference between region of flow and volume

```
g<- ggplot(TrafficFirstGoal,aes(Region,allVolume,fill = factor (AADFYear)))
g+geom_bar(stat = "identity")+theme_bw()+labs(title ="the differance between region of flow and volume
" )+p
```

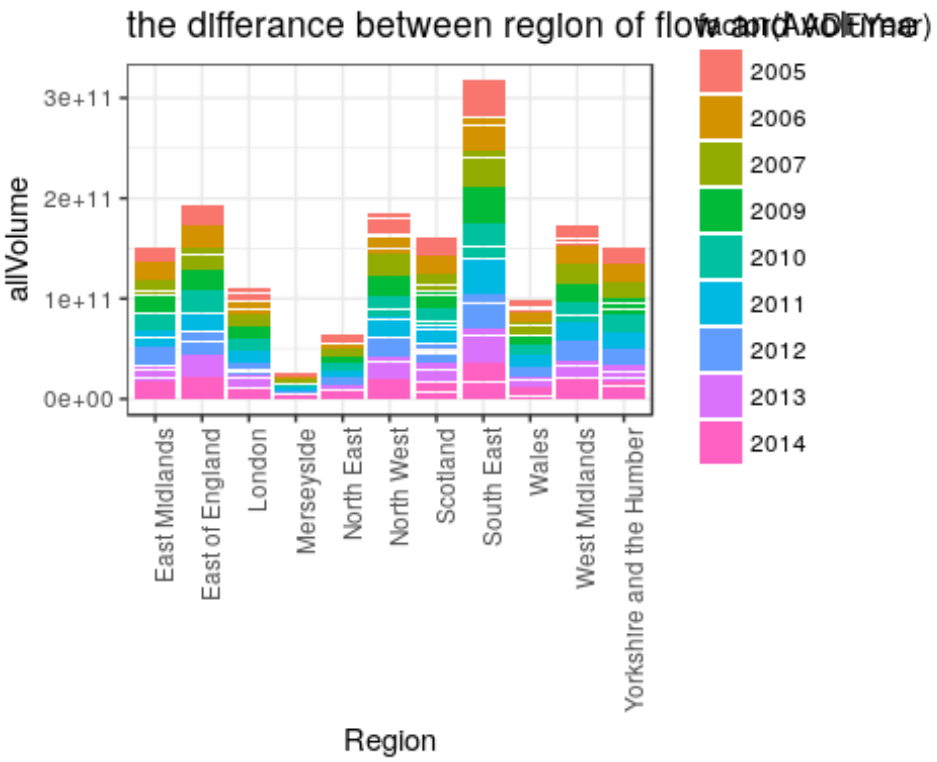


FIGURE7 34: THE DIFFERANCE BETWEEN REGION OF FLOW AND VOLUME

7.2 .3 Third problem

Problem:

Differences between England, Scotland, and Wales first we load the libraries which we need

```
library(dplyr)
library(tidyr)
library(ggplot2)
```

7.2 .3.1 Filter the needed variables

```
trafficNinthGoal <- TrafficAADF %>%
  filter(Region=="Wales"| Region=="Scotland" | Region=="East of England")%>%
  group_by(Region)

head(trafficNinthGoal)
```

```
## # A tibble: 6 x 42
## # Groups:   Region [1]
##   AADFYear    CP Estimation_method Estimation_metho… Region LocalAuthority
##      <int> <int> <chr>            <chr>              <chr>  <chr>
## 1     2000   501 Counted          Manual count       Wales  Newport
## 2     2000   502 Counted          Manual count       Wales  Bridgend
## 3     2000   503 Counted          Manual count       Wales  Swansea
## 4     2000   504 Counted          Manual count       Wales  Isle of Angle…
## 5     2000   505 Estimated        Estimated using … Wales  Gwynedd
```

```
## 6    2000   506 Estimated        Estimated using … Wales  Conwy
## # ... with 36 more variables: Road <chr>, RoadCategory <chr>,
## #   Easting <int>, Northing <int>, StartJunction <chr>, EndJunction <chr>,
## #   LinkLength_km <dbl>, LinkLength_miles <dbl>, PedalCycles <int>,
## #   Motorcycles <int>, CarsTaxis <int>, BusesCoaches <int>,
## #   LightGoodsVehicles <int>, V2AxleRigidHGV <int>, V3AxleRigidHGV <int>,
## #   V4or5AxleRigidHGV <int>, V3or4AxleArticHGV <int>,
## #   V5AxleArticHGV <int>, V6orMoreAxleArticHGV <int>, AllHGVs <int>,
## #   AllMotorVehicles <int>, Lat <dbl>, Lon <dbl>, PedalCyclesVolume <dbl>,
## #   MotorcyclesVolume <dbl>, BusesCoachesVolume <dbl>,
## #   LightGoodsVehiclesVolume <dbl>, V2AxleRigidHGVVolume <dbl>,
## #   V3AxleRigidHGVVolume <dbl>, V3or4AxleArticHGVVolume <dbl>,
## #   V4or5AxleRigidHGVVolume <dbl>, CarsTaxisVolume <dbl>,
## #   V5AxleArticHGVVolume <dbl>, V6orMoreAxleArticHGVVolume <dbl>,
## #   allVehicles <int>, allVolume <dbl>
```

7.2 .3.2 Average of all vehicles in each region

```
average_Vehicles <- summarise(trafficNinthGoal,
                              aveg_pedal= mean(PedalCyclesVolume),
                              aveg_motor = mean(MotorcyclesVolume),
                              aveg_buses = mean(BusesCoachesVolume),
                              aveg_Taxis = mean(CarsTaxisVolume),
                              aveg_Light = mean(LightGoodsVehiclesVolume),
                              aveg_V2 = mean(V2AxleRigidHGVVolume),
                              aveg_v3 = mean(V3AxleRigidHGVVolume),
                              aveg_v34 = mean(V3or4AxleArticHGVVolume),
                              aveg_v45 = mean(V4or5AxleRigidHGVVolume),
                              aveg_v5 = mean(V5AxleArticHGVVolume),
                              aveg_v6 = mean(V6orMoreAxleArticHGVVolume)
)
head(average_Vehicles)
```

```
## # A tibble: 3 x 12
##   Region    aveg_pedal aveg_motor aveg_buses aveg_Taxis aveg_Light aveg_V2
##   <chr>        <dbl>     <dbl>     <dbl>     <dbl>     <dbl>   <dbl>
## 1 East of …    29557.    110372.    84984.  11440071.   1944278. 403246.
## 2 Scotland     11588.    49755.    92119.   6111171.    999780. 229304.
## 3 Wales        11506.    64330.    77230.   7711184.   1219185. 211155.
## # ... with 5 more variables: aveg_v3 <dbl>, aveg_v34 <dbl>,
## #   aveg_v45 <dbl>, aveg_v5 <dbl>, aveg_v6 <dbl>
```

7.2 .3.3 Gather the data

```
average_Vehicles <-gather(average_Vehicles,average,ValueofAverage,-Region )
head(average_Vehicles)
```

```
## # A tibble: 6 x 3
##   Region        average    ValueofAverage
```

```
##   <chr>          <chr>             <dbl>
## 1 East of England aveg_pedal        29557.
## 2 Scotland        aveg_pedal        11588.
## 3 Wales           aveg_pedal        11506.
## 4 East of England aveg_motor       110372.
## 5 Scotland        aveg_motor        49755.
## 6 Wales           aveg_motor        64330.
```

7.2 .3.4 Average of each vehicles in regions

```
g<-ggplot(average_Vehicles,aes(average,ValueofAverage,color = Region))
g+geom_point()+labs(title = "the average of each vehicles in regions")+p
```

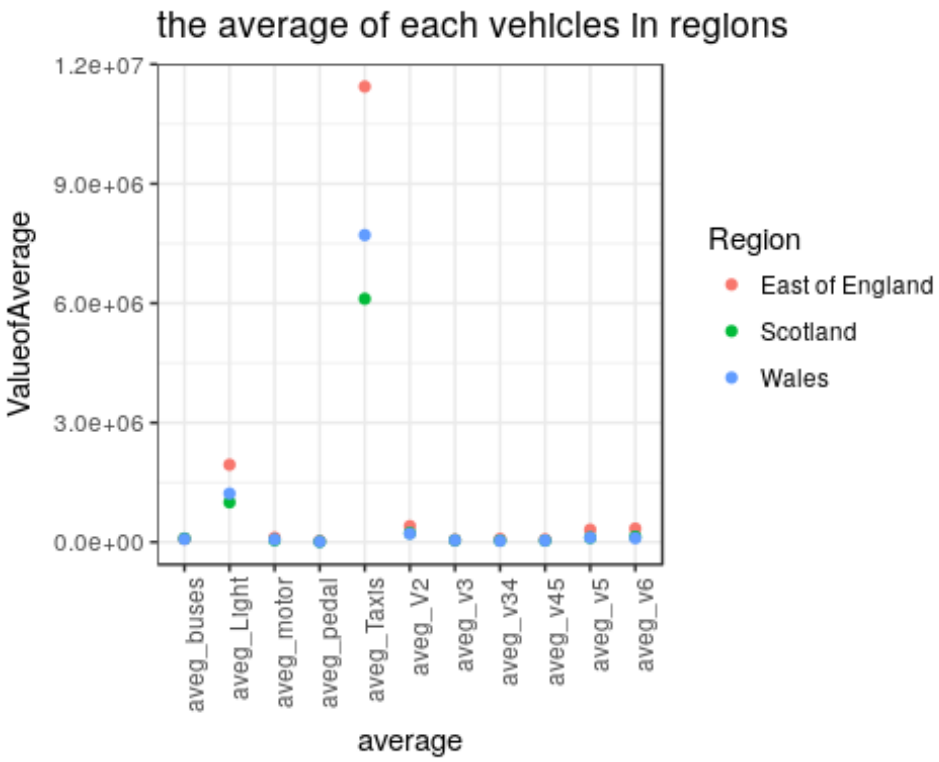


FIGURE7 35: THE AVERAGE OF EACH VEHICLES IN REGIONS

7.2 .3.5 Average of vehicles in all UK and compare which of this region increase of this mean

```
global_average <- mean(TrafficAADF$allVolume)
head(global_average)
```

```
## [1] 11005156
```

7.2 .3.6 Search which region is more busy flow

```
g<-ggplot(trafficNinthGoal,aes(Region,allVolume,color = factor (AADFYear)))
g+geom_point()+theme_bw()+geom_hline(yintercept = global_average)+labs(title = "flow of each region")
```

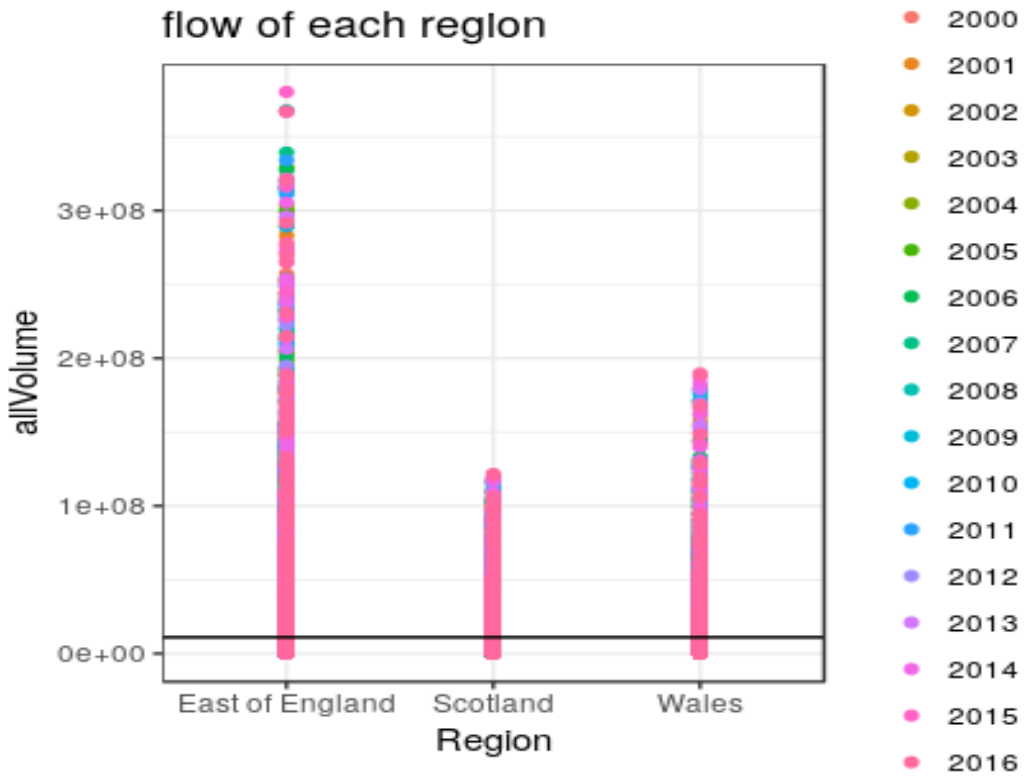


FIGURE7 36: FLOW OF EACH REGION

We see that the england is the first area between them, then wales then scotland and all region consider overcome global average pass the global average over all years old and modern and less over year.

7.2 .3.7 Search in data what difference between them in vehicles

7.2 .3.7.1 Start with the pedal cycles

```
g <- ggplot(trafficNinthGoal,aes(Region,PedalCyclesVolume,color = AADFYear))
g+geom_point()+theme_bw()+geom_hline(yintercept = global_average)+labs(title = "differance between areas in pedal cycles")
```

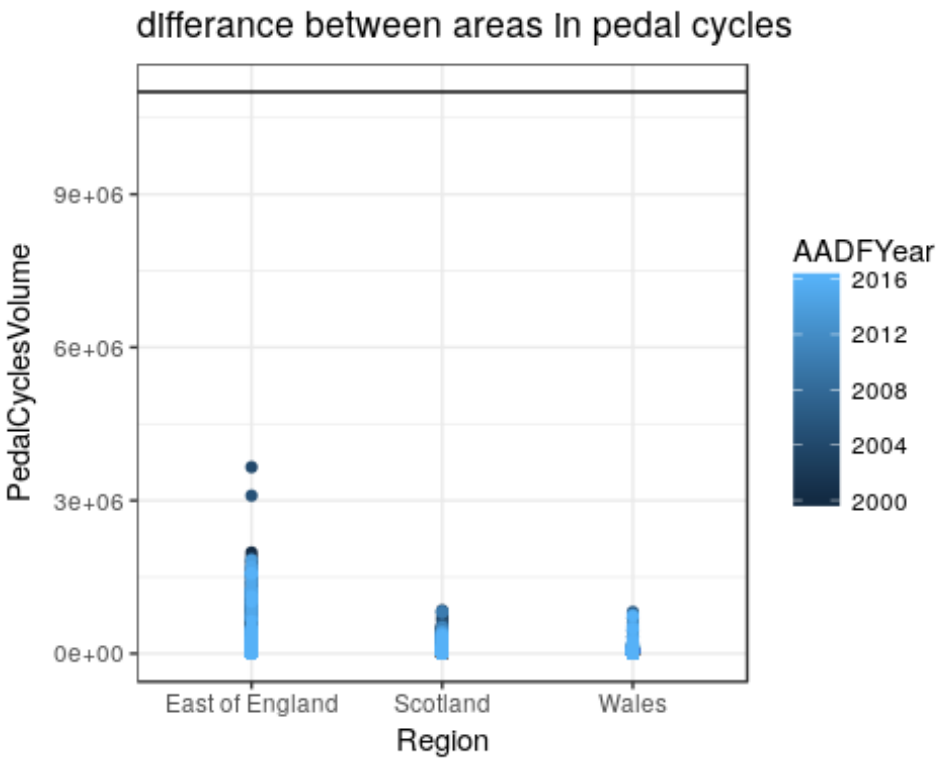


FIGURE7 37:DIFFERANCE BETWEEN AREAS IN PEDAL CYCLES

We see that Englan is uses pedal cycles in the bast is large and be the first country uses of pedal cycles, Scotland and Wales are equal.

7.2 .3.7.2 Difference in motor cycles

```
g <- ggplot(trafficNinthGoal,aes(Region,MotorcyclesVolume,color = AADFYear))
g+geom_point()+theme_bw()+geom_hline(yintercept = global_average)+labs(title = "differance between areas in motor cycles")
```

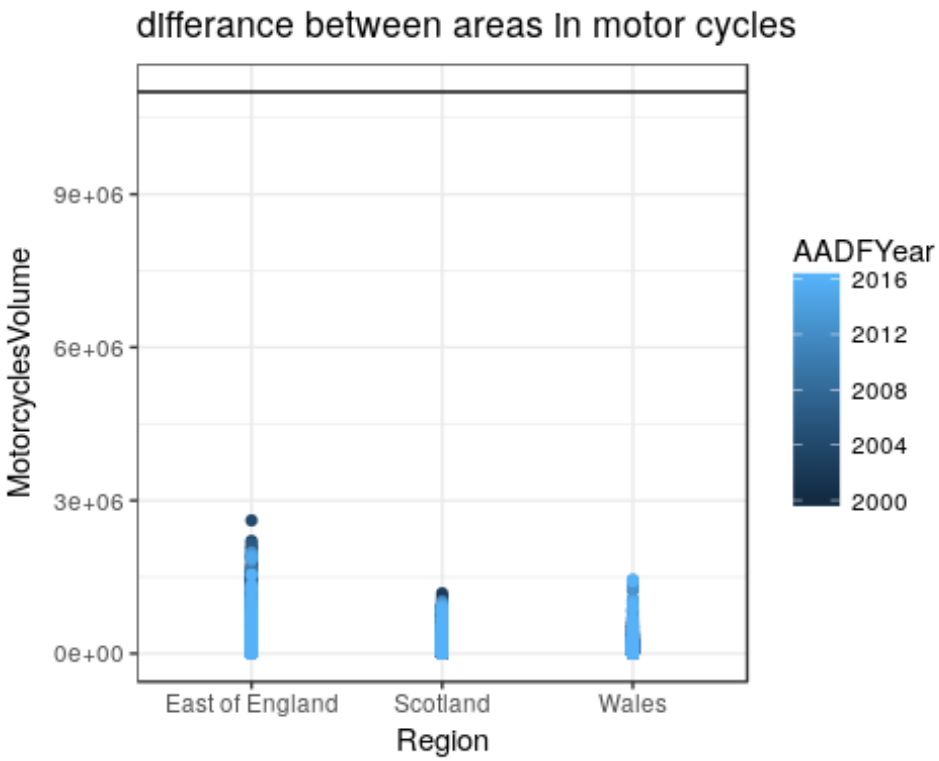


FIGURE7 38: DIFFERANCE BETWEEN AREAS IN MOTOR CYCLES

England take the first place using the motor cycles in the old years and England be in first blace using motor cycles.

7.2 .3.7.3 we see the difference in BusesCoaches

```
g <- ggplot(trafficNinthGoal,aes(Region,BusesCoachesVolume,color = AADFYear))
g+geom_point()+theme_bw()+geom_hline(yintercept = global_average)+labs(title = "differance between areas in buses coaches")
```

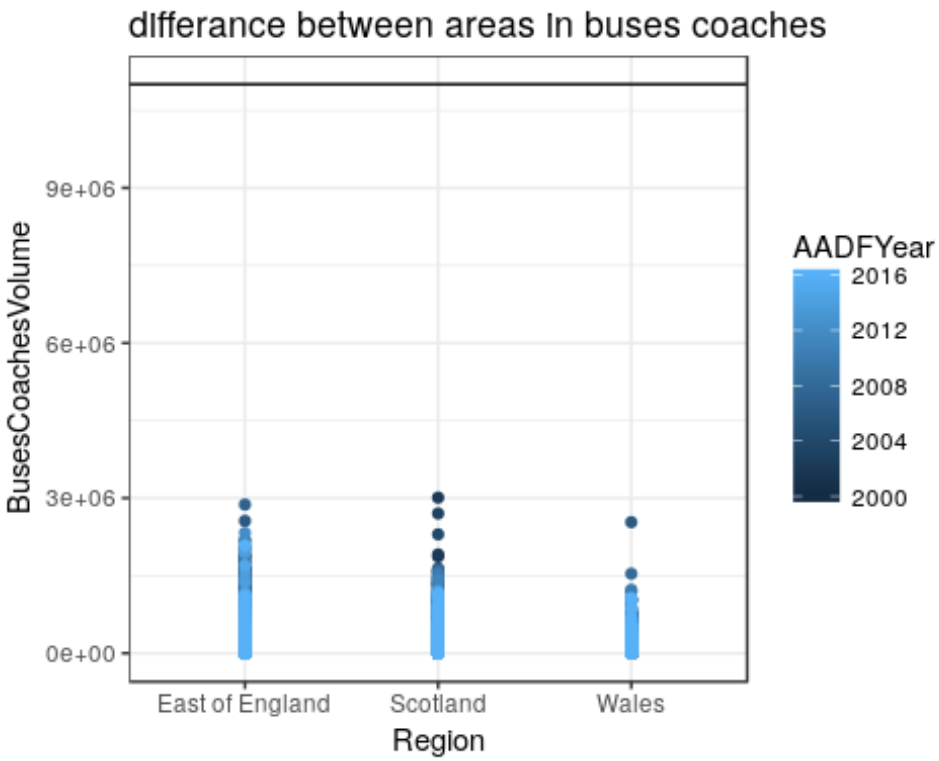


FIGURE7 39: DIFFERANCE BETWEEN AREAS IN BUSES COACHES

Scotland in the first place using buses then wales then england not pass the global average.

7.2 .3.7.4 Difference between car taxis

```
g <- ggplot(trafficNinthGoal,aes(Region,CarsTaxisVolume,color = AADFYear))
g+geom_point()+theme_bw()+geom_hline(yintercept = global_average)+labs(title = "differance between areas in taxis")
```

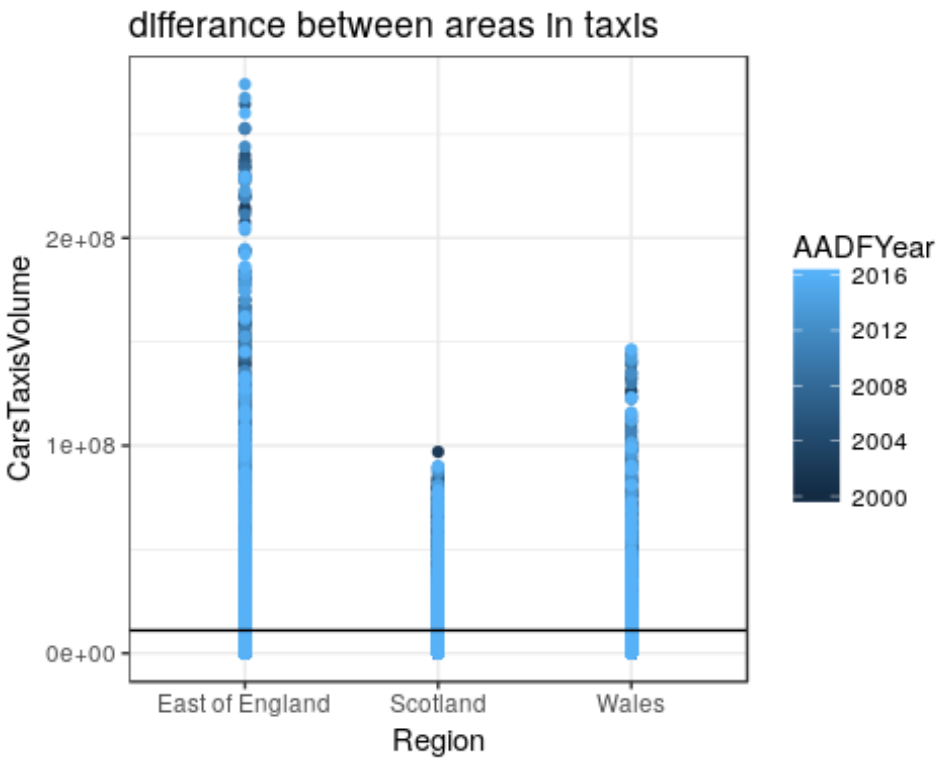


FIGURE7 40: DIFFERANCE BETWEEN AREAS IN TAXIS

Pass the global average in all regions England is in the first place in use then Wales then Scotland.

7.2 .3.7.5 Difference between light goods vehicles

```
g <- ggplot(trafficNinthGoal,aes(Region,LightGoodsVehiclesVolume,color = AADFYear))
g+geom_point()+theme_bw()+geom_hline(yintercept = global_average)+labs(title = "differance between areas in light goods")
```

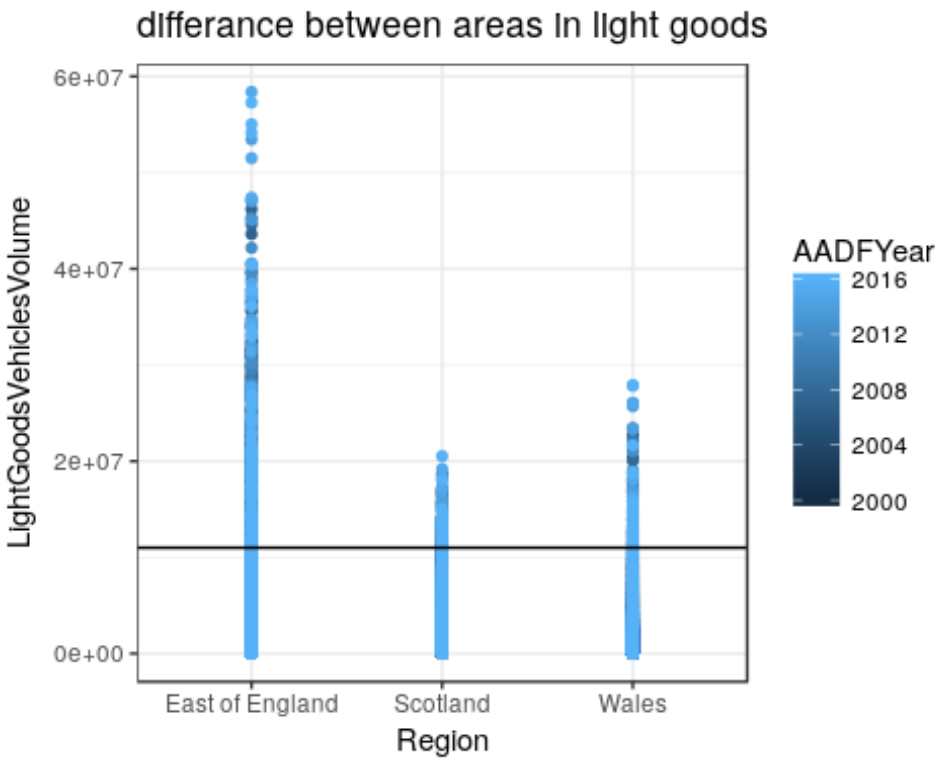


FIGURE7 41: DIFFERANCE BETWEEN AREAS IN LIGHT GOODS

Pass the global average England in first place then Wales then Scotland.

7.2 .3.7.6 we see the differace between V2AxleRigidHGV

```
g <- ggplot(trafficNinthGoal,aes(Region,V2AxleRigidHGVVolume,color = AADFYear))
g+geom_point()+theme_bw()+geom_hline(yintercept = global_average)+labs(title = "differance between areas in V2AxleRigidHGV")
```

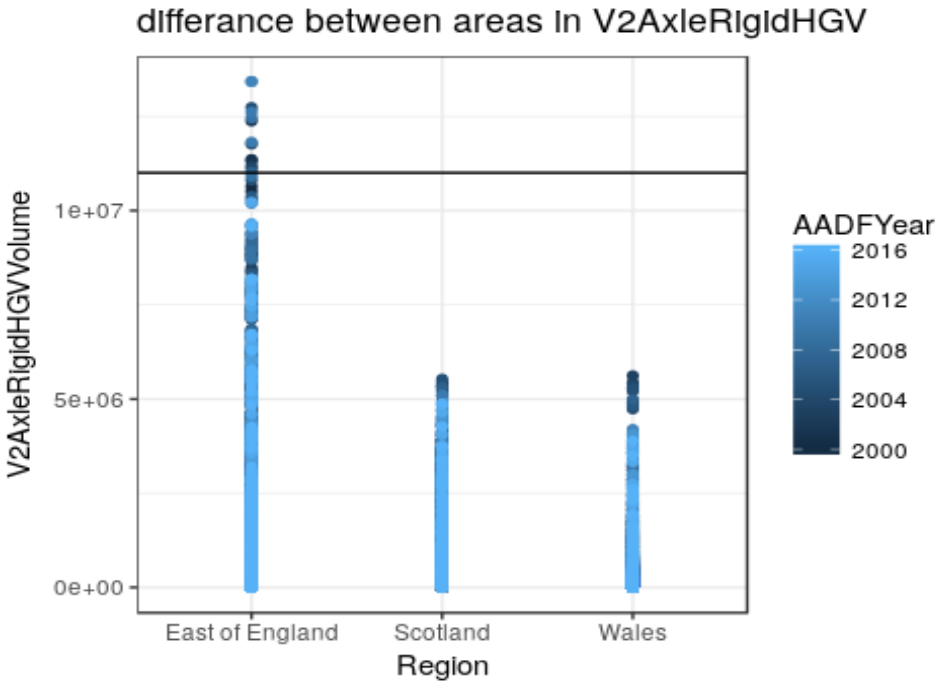


FIGURE7 42: DIFFERANCE BETWEEN AREAS IN V2AXLERIGIDHGV

Pass the global average only in England.

7.2 .3.7.7 we see the difference between V3AxleRigidHGV

```
g <- ggplot(trafficNinthGoal,aes(Region,V3AxleRigidHGVVolume,color = AADFYear))
g+geom_point()+theme_bw()+geom_hline(yintercept = global_average)+labs(title = "differance between areas in V3AxleRigidHGV")
```

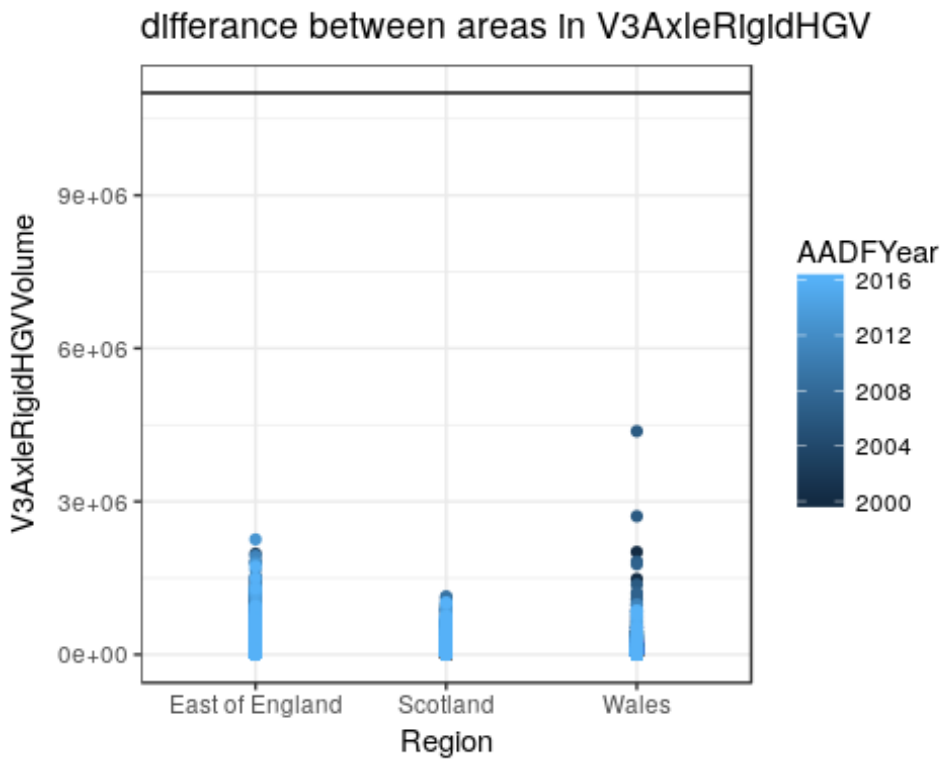


FIGURE7 43: DIFFERANCE BETWEEN AREAS IN V3AXLERIGIDHGV

In the first place Wales then England then Scotland and uses in Scotland less over years.

7.2 .3.7.8 Difference between V3or4AxleArticHGV

```
g <- ggplot(trafficNinthGoal,aes(Region,V3or4AxleArticHGVVolume,color = AADFYear))
g+geom_point()+theme_bw()+geom_hline(yintercept = global_average)+labs(title = "differance between areas in V3or4AxleRigidHGV")
```

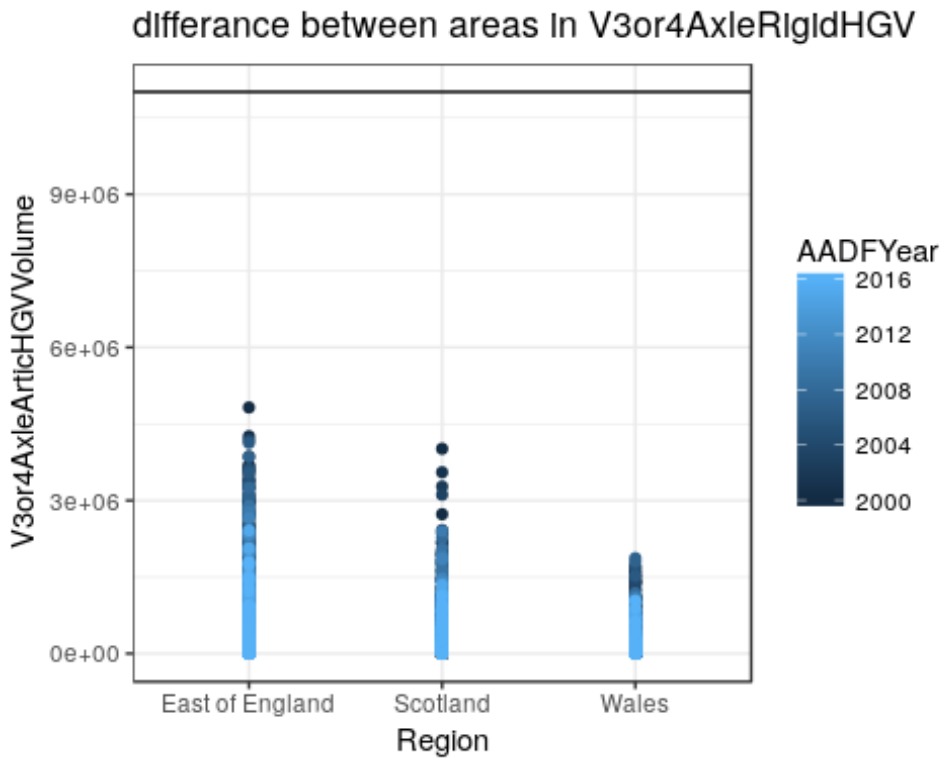


FIGURE7 44: DIFFERANCE BETWEEN AREAS IN V3OR4AXLERIGIDHGV

Less uses over years and England in the first place then Scotland then Wales.

7.2 .3.7.9 Difference between V4or5AxleRigidHGV

```
g <- ggplot(trafficNinthGoal,aes(Region,V4or5AxleRigidHGVVolume,color = AADFYear))
g+geom_point()+theme_bw()+geom_hline(yintercept = global_average)+labs(title = "differance between areas in V4or5AxleRigidHGV")
```

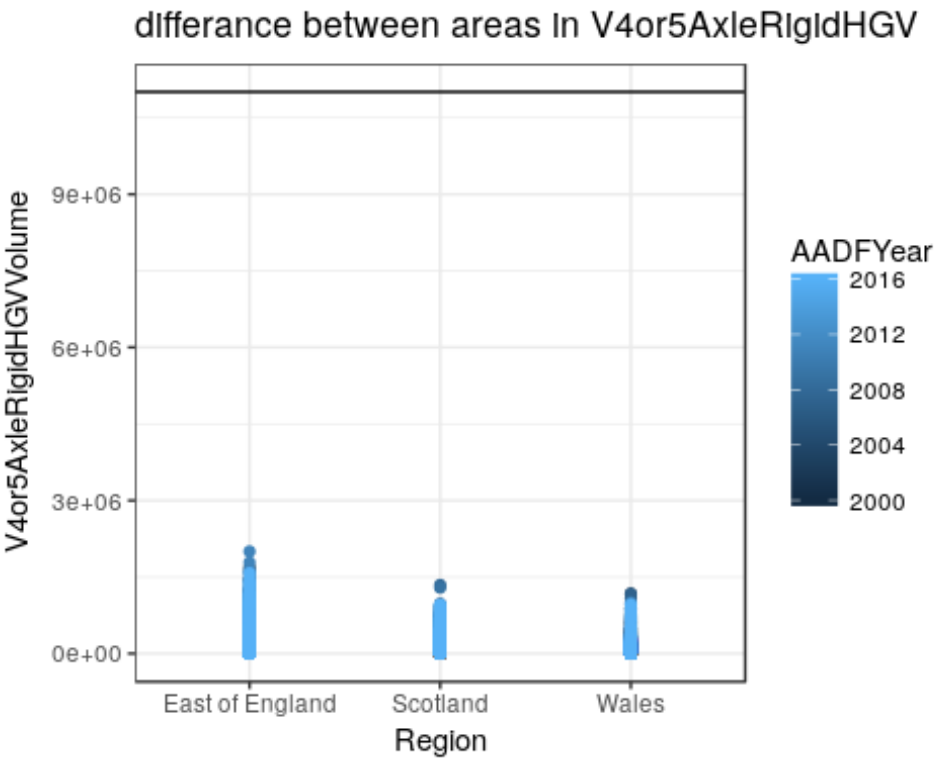


FIGURE7 45:DIFFERANCE BETWEEN AREAS IN V4OR5AXLERIGIDHGV

More uses in England then Scotland then Wales.

7.2 .3.7.10 Difference between V5AxleArticHGV

```
g <- ggplot(trafficNinthGoal,aes(Region,V5AxleArticHGVVolume,color = AADFYear))
g+geom_point()+theme_bw()+geom_hline(yintercept = global_average)+labs(title = "differance between areas in V5AxleRigidHGV")
```

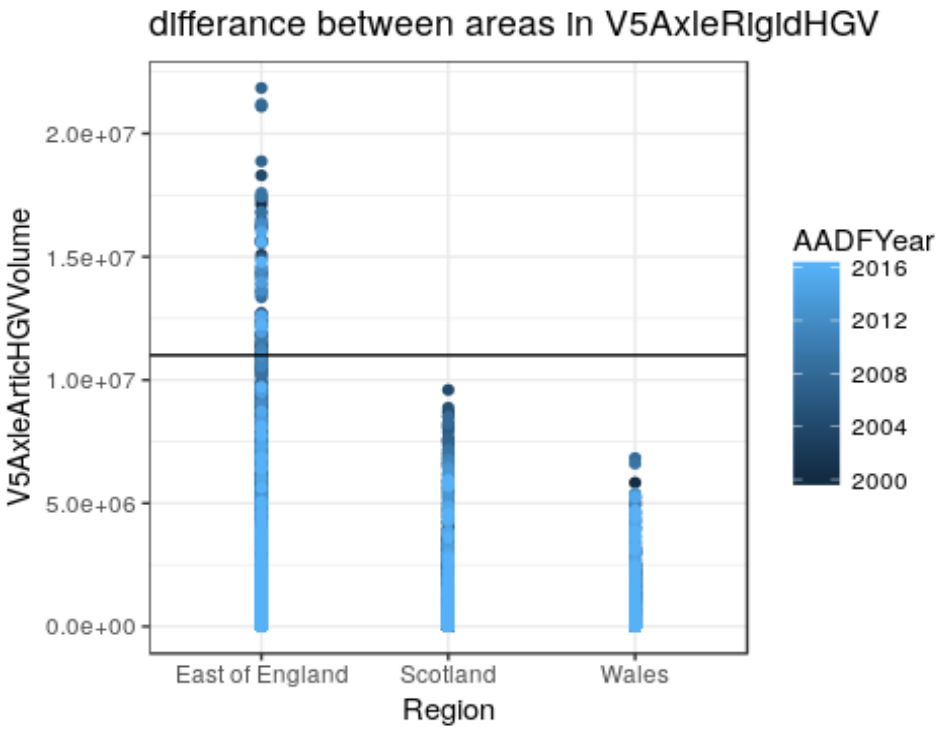


FIGURE7 46:DIFFERANCE BETWEEN AREAS IN V5AXLERIGIDHGV

England passes the global average and less uses over years then Scotland then Wales.

7.2 .3.7.11 Difference between V6orMoreAxleArticHGV

```
g <- ggplot(trafficNinthGoal,aes(Region,V6orMoreAxleArticHGVVolume,color = AADFYear))
g+geom_point()+theme_bw()+geom_hline(yintercept = global_average)+labs(title = "differance between areas in V6ormoreAxleRigidHGV")
```

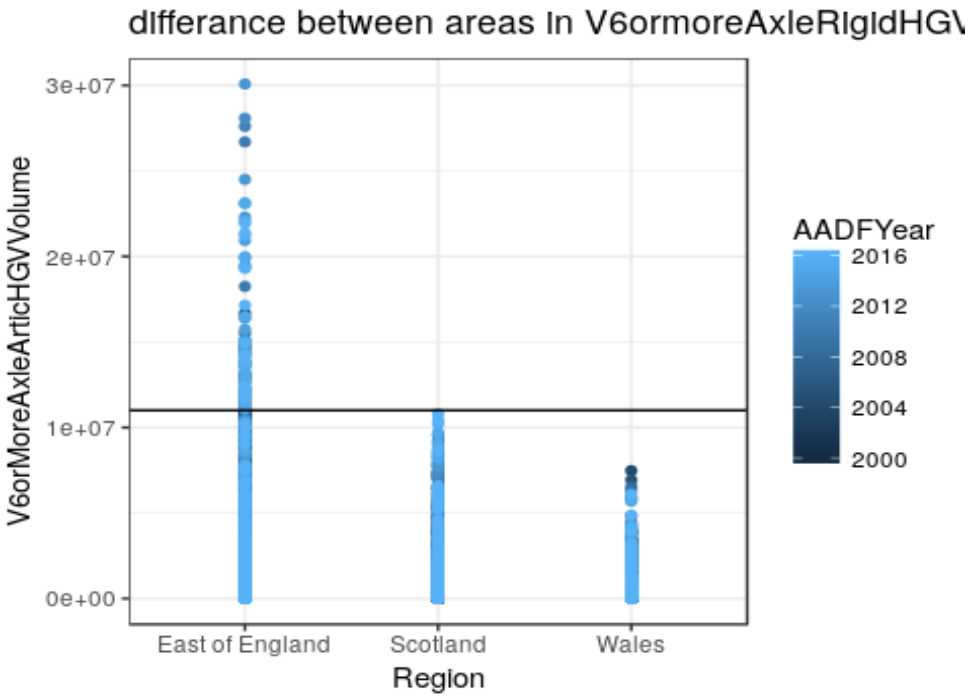


FIGURE7 47:DIFFERANCE BETWEEN AREAS IN V6ORMOREAXLERIGIDHGV

England passes the global average then Scotland then Wale

7.2.4 Fourth problem "How to improve accident Rate? ".

7.2.4.1-Show relation between road type and condition affect on it.

Discovering the relation between speed limit and number of accidents.

```
ggplot(data = accidents) +
 geom_bar(mapping = aes(x =Speed_limit, fill =
factor(Road_Type)),position="dodge",width=8)+scale_x_discrete(limits=c(10,20,30,40,50,60,70))+

 labs(

  title="discovering the relation betwwen  speed limit number of accidents",
  fill="Road type",
  y="numer of accidents",
  x="speed limit"

 )
```

```
## Warning: position_dodge requires non-overlapping x intervals
```

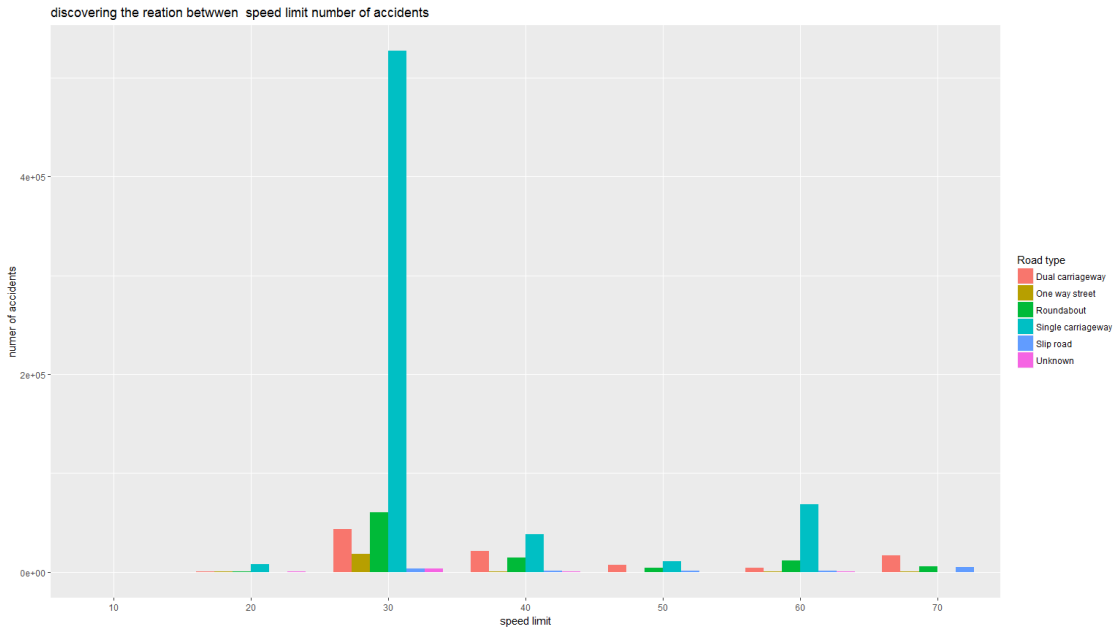


FIGURE7 48:DISCOVERING THE RELATION BETWWEN SPEED LIMIT NUMBER OF ACCIDENTS

-From the figure we find that the highest number of accidents at speed 30 and road type is single carrageaway and we can deduces that using single carriageway increasing rate of traffic so lead to accidents

7.2.4.2 Discovering the relation between speed and number of accidents.

Group accident according to speed rate.

```
spee<-group_by(allacc,Speed_limit)
head(spee)
```

```
## # A tibble: 6 x 28
## # Groups:   Speed_limit [1]
##       X AccidentID  Location_Easting~ Location_Northin~ Longitude Latitude
##   <int> <fct>               <int>          <int>    <dbl>    <dbl>
## 1     2 200501BS00~         524170         181650   -0.212     51.5
## 2     7 200501BS00~         524220         180830   -0.211     51.5
## 3     9 200501BS00~         527350         177650   -0.167     51.5
## 4    10 200501BS00~         524550         180810   -0.207     51.5
## 5    11 200501BS00~         526240         178900   -0.183     51.5
## 6    12 200501BS00~         526170         177690   -0.184     51.5
## # ... with 22 more variables: Police_Force <int>, Accident_Severity
```

Summarize to get the number of accidents according to speed limit.

```
spee1<-dplyr::summarize(spee,spee_acci=n())
```

Plot the relation between Speed limit and number of accidents.

 # by speed:2

```
ggplot(data =spee1,mapping = aes(x =Speed_limit, y=spee_acci)) +
 geom_point(color="red",size=5)+geom_smooth(se = FALSE,method = 'loess',span=0.7)+

 labs(

  title=" discovering the reation between  speed and number of accidents  ",
  x=" speed_limit",
  y="numer of accidents",
  fill="Days"

 )+geom_text(aes(label=spee_acci),

          vjust = -0.7, size =5,color="blue")+

 theme_minimal()
```

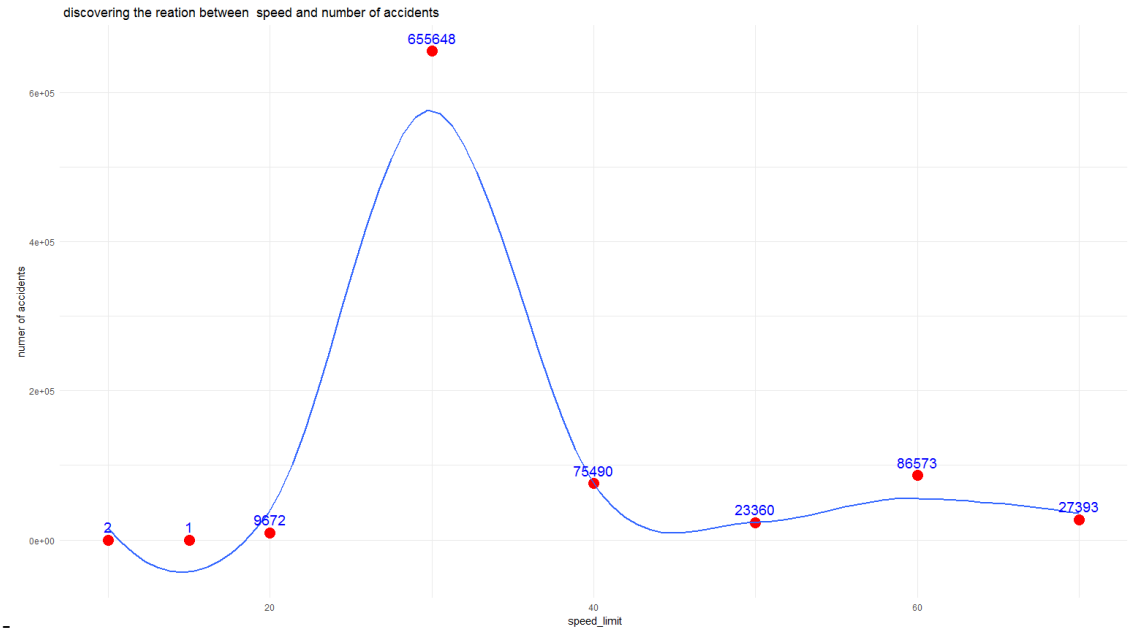


FIGURE7 49: DISCOVERING THE REATION BETWEEN SPEED AND NUMBER OF ACCIDENTS

- From the figure we found that the highest number of accidents which is 655648 at speed 30

7.2.4.3 Discovering the relation between day of week and number of accidents.

Group accidents by days of week.

```
#group by  days of week:
day<-group_by(allacc,Day_of_Week)
head(day)
```

```
## # A tibble: 6 x 28
## # Groups:   Day_of_Week [5]
##       X AccidentID  Location_Easting~ Location_Northin~ Longitude Latitude
##   <int> <fct>               <int>          <int>    <dbl>    <dbl>
## 1     2 200501BS00~         524170         181650   -0.212     51.5
## 2     7 200501BS00~         524220         180830   -0.211     51.5
## 3     9 200501BS00~         527350         177650   -0.167     51.5
## 4    10 200501BS00~         524550         180810   -0.207     51.5
## 5    11 200501BS00~         526240         178900   -0.183     51.5
## 6    12 200501BS00~         526170         177690   -0.184     51.5
## # ... with 22 more variables: Police_Force <int>, Accident_Severity <fct>,
```

Summarize accidents according to day of

```
# summarize by  by  days of week:1
 dayweek<-dplyr::summarize(day,accident_per_dayofweek=n())
```

Compute percentage of accidents according to day of week.

```
# compute percentage.
dayweek<-mutate(dayweek,percent=paste0(round(accident_per_dayofweek/nrow(day)*100,1),"%"))

ggplot(accidents) +
 geom_bar(aes(x=Day_of_Week, fill = factor(Road_Type)),position="dodge")+

 labs(
```

Plot: the relation between day of week and number of accidents.

```
  title=" discovering the relation between day of week and number of accidents ",
  fill="Road type",
  y="number of accidents",
  x="Days"

)
```

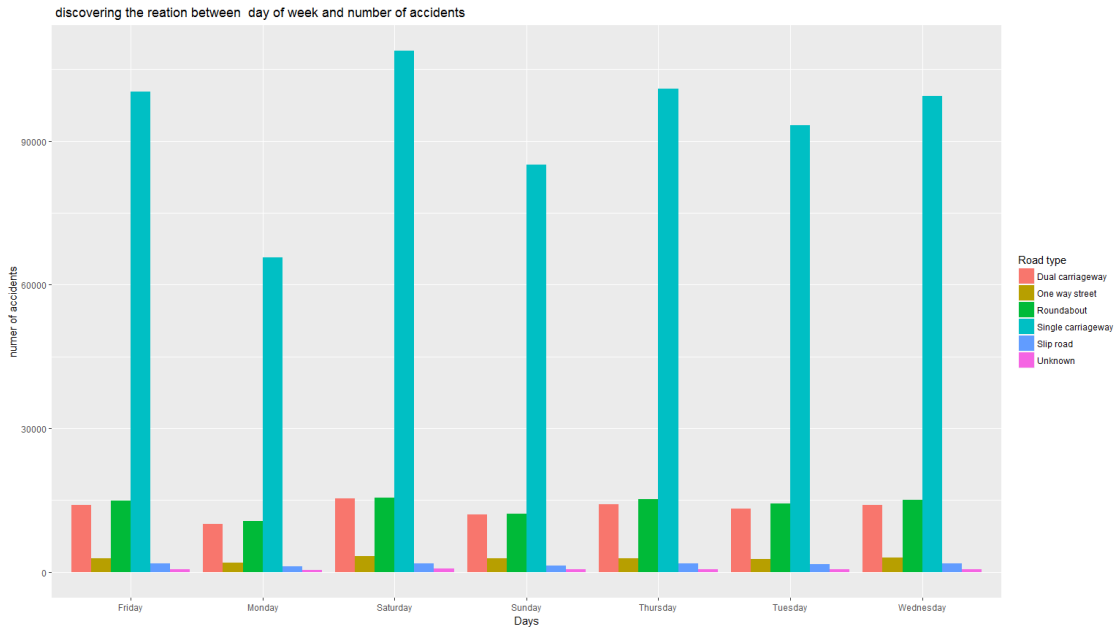


FIGURE7 50: DISCOVERING THE RELATION BETWEEN DAY OF WEEK AND NUMBER OF ACCIDENTS

- From the figure we found that the number of accidents the highest at Saturday at single carriageway

7.2.4.4 Discovering the relation between day of week and number of accidents.

 # by day of week:2

```
ggplot(data =dayweek,mapping = aes(x =dayweek$Day_of_Week,
y=accident_per_dayofweek,color=Day_of_Week)) +
 geom_point(size=5)+geom_smooth(se = FALSE,method = 'loess')+

 labs(

  title=" discovering the reation between  day of week and number of accidents  ",
  x=" Days",
  y="numer of accidents",
  color="Days"

 )+geom_text(aes(label=percent),
           vjust = 1.6, size =5,color="blue")+

theme_minimal()
```

FIGURE7 51:DISCOVERING THE REATION BETWEEN DAY OF WEEK AND NUMBER OF ACCIDENTS

-This figure show the highest percentage of accidents at Saturday of week

7.2.4.5 Relation between speed limit and number of acidents in rural and Urban area.

#relation between speed limit and number of acidents in rural and Urban area.

```
ggplot(data = accidents) +
 geom_bar(mapping = aes(x = accidents$Road_Type, fill =
factor(accidents$Speed_limit)),position="dodge")+

 labs(

  title=" discovering the reation between  speed limit and  number of accidents  ",
  subtitle="differnce between Urban and Rural area",
  x="Road type",
  y="numer of accidents",
  fill="speed limit"

 )+facet_wrap(~accidents$Urban_or_Rural_Area)+theme_bw()
```

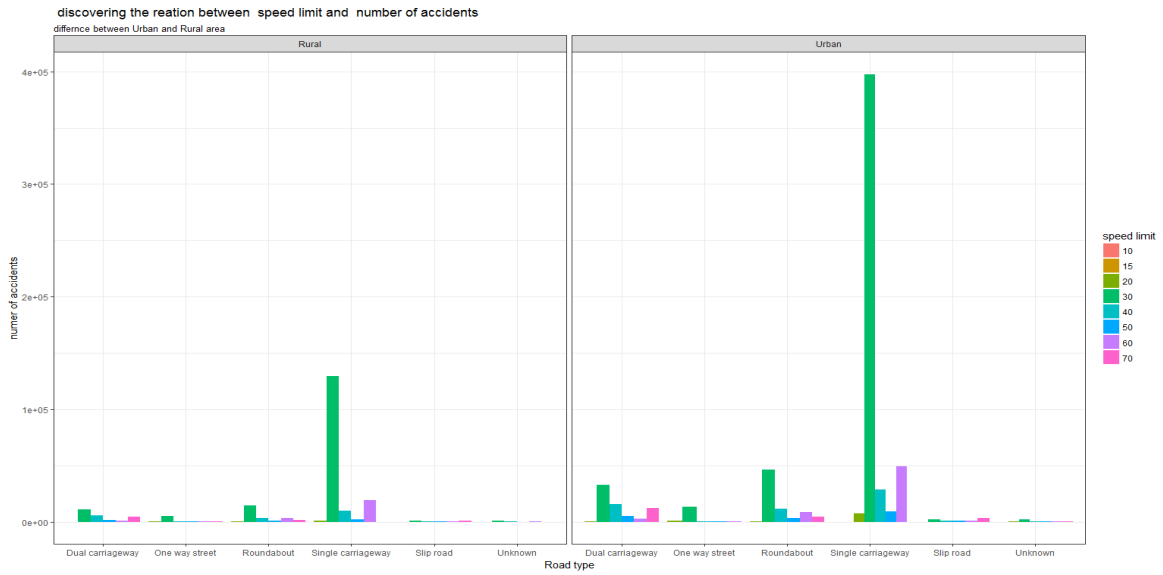


FIGURE7 52:DISCOVERING THE REATION BETWEEN SPEED LIMIT AND NUMBER OF ACCIDENTS

This figure show:

That the highest number of accidents at Saturday, speed is “30”, and single carriageway in urban area as urban area is most increasing with trafficflow.

7.2.4.6 Discovering the relation between light conditions and the number of accidents

```
#lights conditions

ggplot(data = accidents) +
 geom_bar(mapping = aes(x = accidents$Road_Type, fill =
factor(accidents$Light_Conditions)),position="dodge")+

 labs(

  title=" discovering the relation between   light conditions  and the number of accidents  ",
  subtitle="difference between Urban and Rural area",
  x="Road type",
  y="number of accidents",
  fill="Light condition"

 )+facet_wrap(~accidents$Urban_or_Rural_Area)+theme_bw()
```

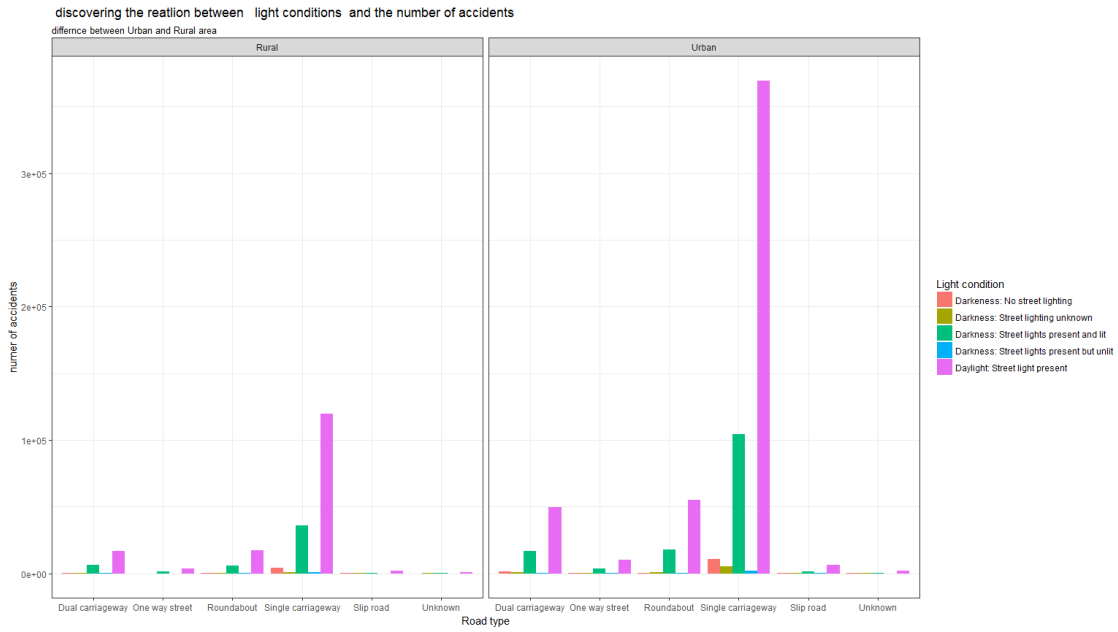


FIGURE7 53:DISCOVERING THE RELATION BETWEEN LIGHT CONDITIONS AND THE NUMBER OF ACCIDENTS

-This figure show that the highest number of accidents at urban area and light condition is day light street light present.

7.2.4.7 Discovering the relation between Weather conditions and the number of accidents.

#weathercondition

```
ggplot(data = accidents) +
 geom_bar(mapping = aes(x = accidents$Road_Type, fill =
factor(accidents$Weather_Conditions)),position="dodge")+

 labs(

  title=" discovering the relation betewen Weather conditions  and the number of accidents  ",
  subtitle="differnce between Urban and Rural area",
  x="Road type",
  y="number of accidents",
  fill="Weather condition condition"

 )+facet_wrap(~accidents$Urban_or_Rural_Area)+theme_bw()
```

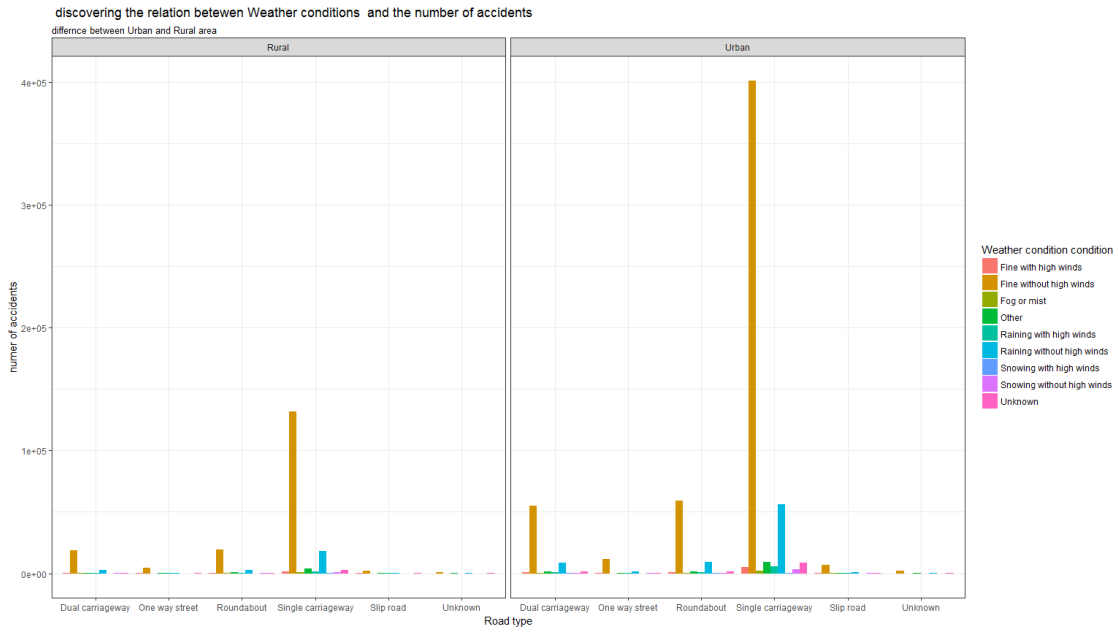


FIGURE7 54:DISCOVERING THE RELATION BETEWEN WEATHER CONDITIONS AND THE NUMBER OF ACCIDENTS

- This figure show that the highest number of accidents at urban area and weather condition is “fine without high winds” we can deduce that people at this weather condition go outdoor so rate of traffic flow at this time is increasing

7.2.4.8 Discovering the relation between Road_surface contion condition and the number of accidents.

We make filter to remove remove zero value.

```
acc<-filter(accidents,!(accidents$Road_Surface_Conditions==0))#remove zero value
```

Plot: the relation between Road_surface contion condition and thenumber of accidents.

```
# Road surface contion



# plot:
ggplot(data = accidents) +
 geom_bar(mapping = aes(x = accidents$Road_Type, fill =
factor(accidents$Road_Surface_Conditions)),position="dodge")+

 labs(

  title=" discovering the relation between Road_surface contion condition  and thenumber of accidents ",
```

```
 subtitle="differnce between Urban and Rural area",
 x="Road type",
 y="numer of accidents",
 fill=" Road surface condition"

)+facet_wrap(~accidents$Urban_or_Rural_Area)+theme_bw()
```

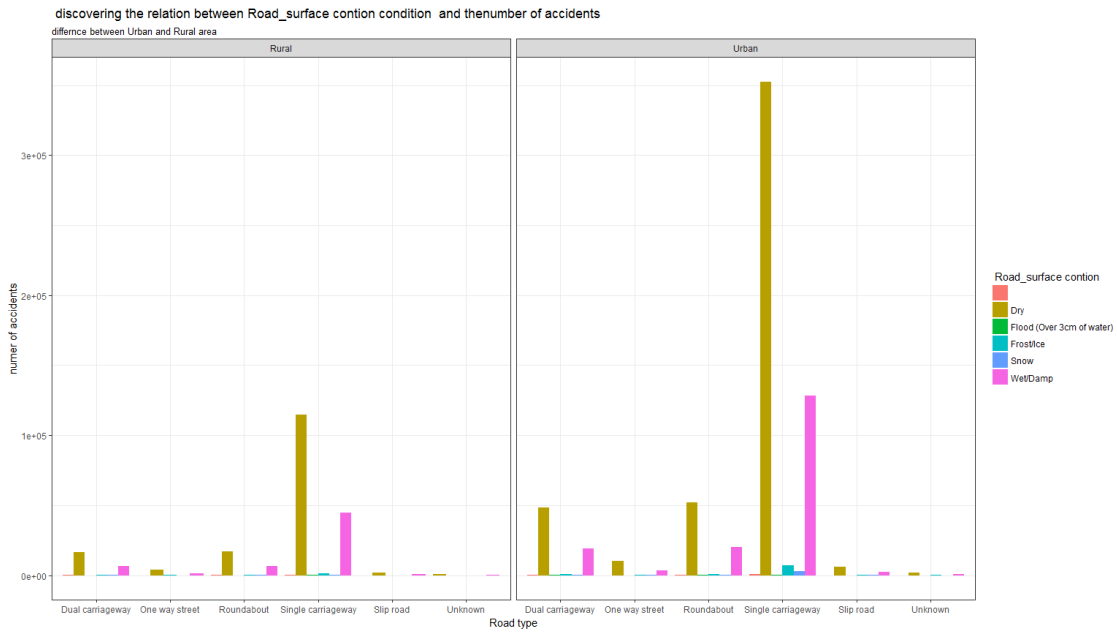


FIGURE7 55:DISCOVERING THE RELATION BETWEEN ROAD_SURFACE CONTION CONDITION AND THENUMBER OF ACCIDENTS

- This figure show that the highest number of accidents at urban area and road surface conditions is “Dry”

7.2.4.8 Road Junction control and the impact of it on the number of accidents on each Road.

```
# junction control contion on each Road

#plot

ggplot(data = accidents) +
 geom_count(mapping = aes(x = accidents$Road_Type, y = accidents$X1st_Road_Class,color = ..n..))+
 facet_wrap(~Junction_Control)+
 labs(
  title="Road Junction control and the impact of it on the number of accidents on each Road",
  x="Road Type",
  y="Road class",
  subtitle="Note:each graph specifiy How the Road junction is controled and impact of that on accident
Rate"

 )
```

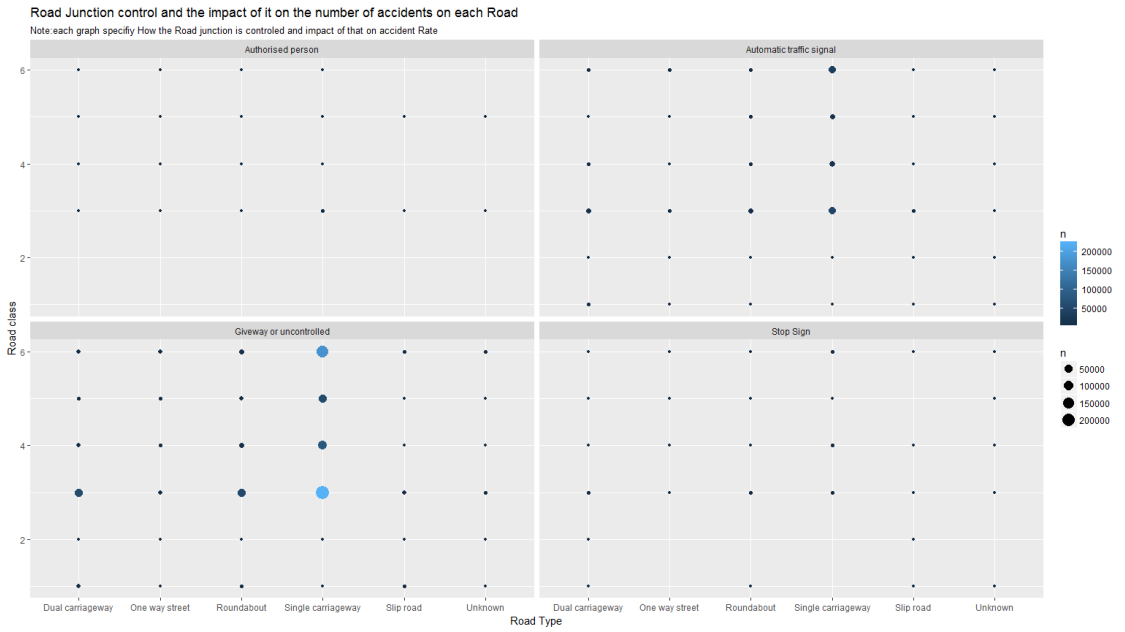


FIGURE7 56:ROAD JUNCTION CONTROL AND THE IMPACT OF IT ON THE NUMBER OF ACCIDENTS ON EACH ROAD

-This figure show

That the highest number of accidents at road type is “Single carriageway”and junction control is” Giveway or uncontrolled”.

7.2.4.9 The relation between Road_Surface_Conditions and Road type according to speed limit.

 # speed and weather contions and road:

```
ggplot(data = accidents) +
 geom_count(mapping = aes( x= Road_Surface_Conditions,y= Road_Type,color = ..n..))+
 facet_wrap(~Speed_limit)+
 labs(
  title="the relation between Road_Surface_Conditions and Road type according to speed limit.",
  y="Road Surface Conditions",
  x=" Road type",
  color="Number of accidents",
  subtitle="Note:each graph specifiy How the speed affect on accients"

 )
```

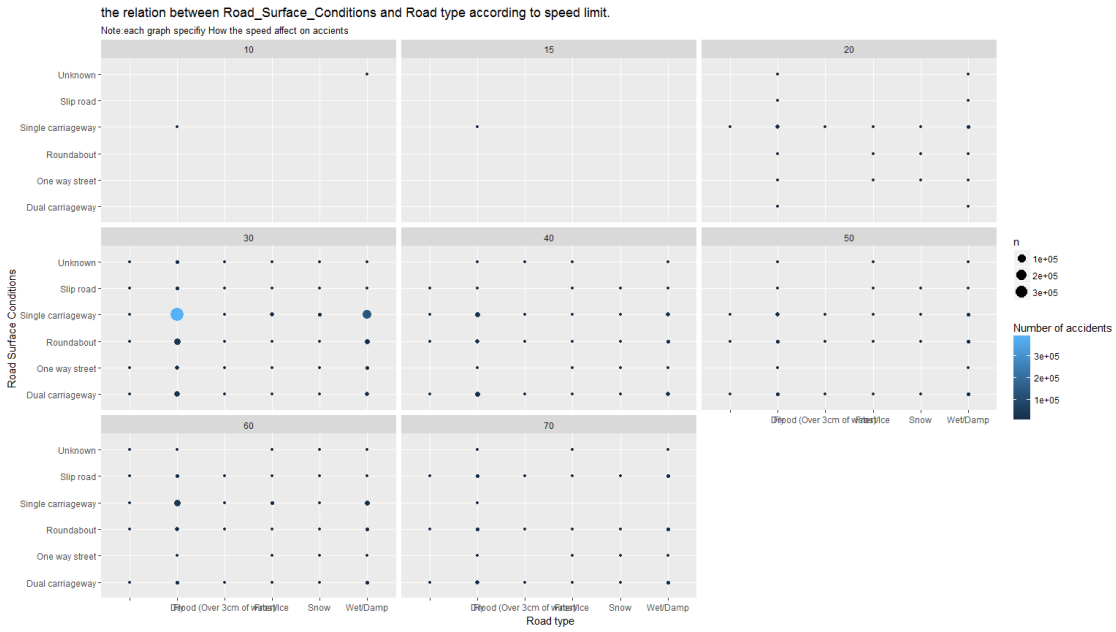


FIGURE7 57:THE RELATION BETWEEN ROAD_SURFACE_CONDITIONS AND ROAD TYPE ACCORDING TO SPEED LIMIT

-This figure show

That the highest number of accidents at “Single carriageway “and road surface condition is “Dry” when speed is “30”.

7.2.4.10 Road and speed limit and weather condition and road surface condition.

 # road and speed limit and weather contion and road surface condtion.

```
ggplot(data=accidents,aes(x=Speed_limit,fill=accidents$Road_Surface_Conditions))+geom_bar(position="dodge",width=10)+
 labs(

   subtitle=" accident contion ",
   title ="Discovering  which affect on accients\nroad and speed limit and weather contion \nand road surface condtion.",
  x="year",
  y=" number of accidents",
  fill="weather conations"
 )+facet_grid(Weather_Conditions~Road_Type)+scale_x_discrete(limits=seq(10,70,by=10))
```

```
## Warning: position_dodge requires non-overlapping x intervals
```

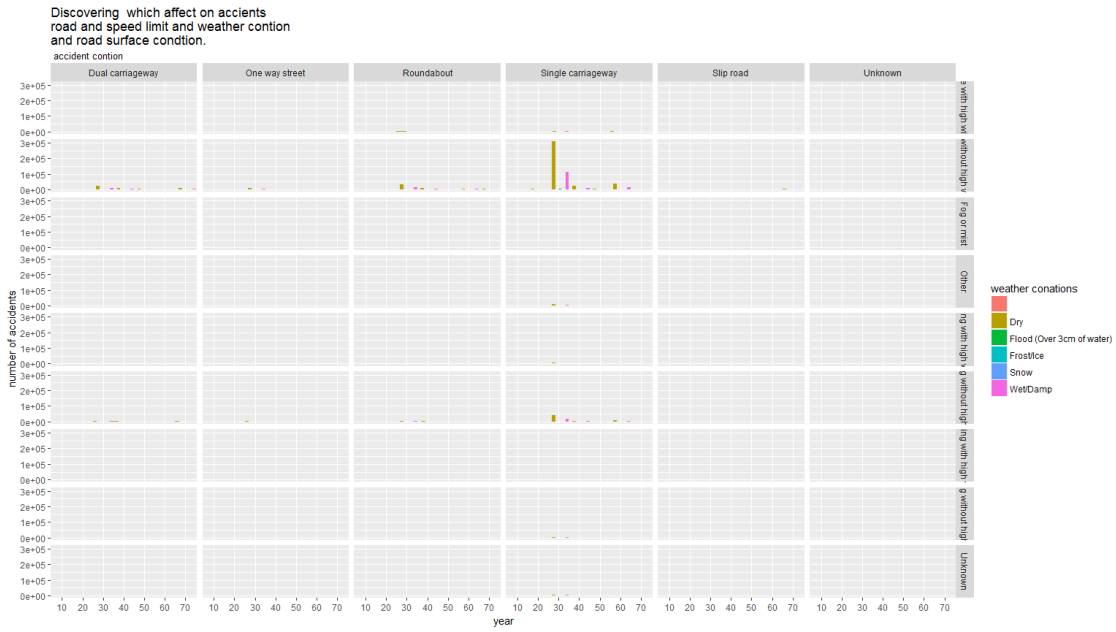


FIGURE7 58:DISCOVERING WHICH AFFECT ON ACCIENTS\NROAD AND SPEED LIMIT AND WEATHER CONTION \NAND ROAD SURFACE CONDTION

-This figure show the highest number of accidents at road type" Single carriageway" and weather condition is "Dry"

7.2.4.11 the relation between Road , speed limit and Human_Control.

```
# road and speed limit and Human_Control
# plot:
ggplot(data=accidents,aes(x=Speed_limit,fill=accidents$Pedestrian_Crossing.Human_Control))+geom_bar(
position="dodge",width=5)+
 labs(

  subtitle=" accident condations ",
  title=" road and speed limit and Human_Control",
 x="speed limit",
 y=" number of accidents",
 fill="Pedestrian Crossing\nHuman_Control"
)+facet_grid(Junction_Control~Road_Type)+scale_x_discrete(limits=seq(10,70,by=10))
```

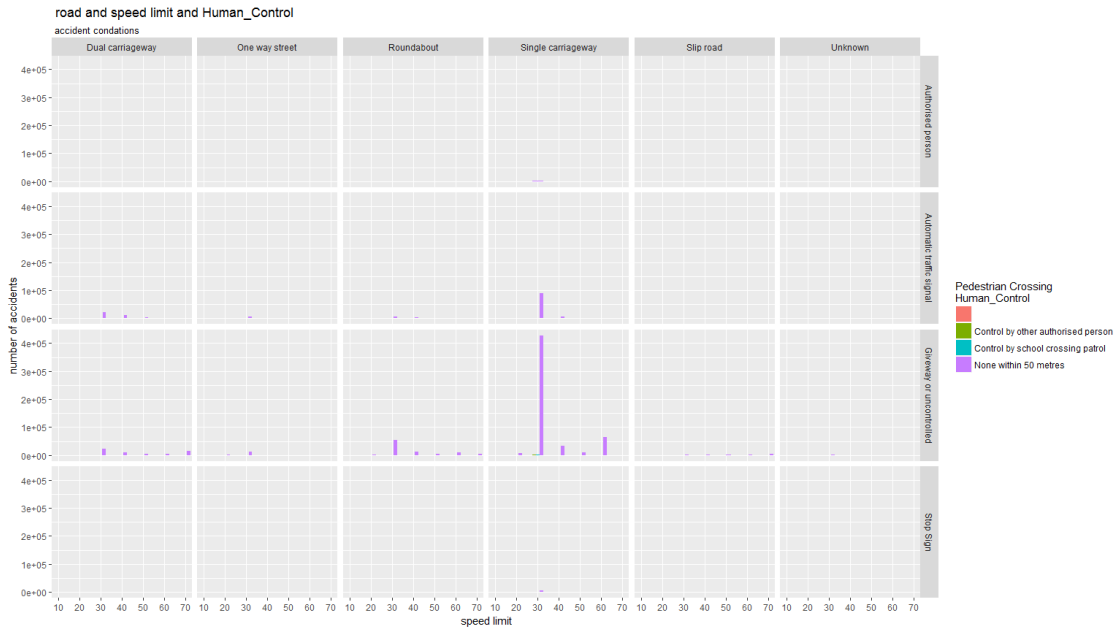


FIGURE7 59:ROAD AND SPEED LIMIT AND HUMAN_CONTROL

-This figure show that the highest number of accidents at road type" Single carriageway" ,speed "30" and Pedestrian Crossing Human Control is "None within 50 metres".

7.2.4.12 Accident status: high or moderate.

 # accident status:high or moderate.

```
bar <- ggplot(data = accidents) +
 geom_bar(
  mapping = aes(x = accidents$Road_Type, fill = Accident_Severity),
  width = 1
 ) +
 theme(aspect.ratio = 1) +
 labs(x = NULL, y = NULL,
     title="accident statues low -high -moderate"
     )

bar + coord_flip()
```

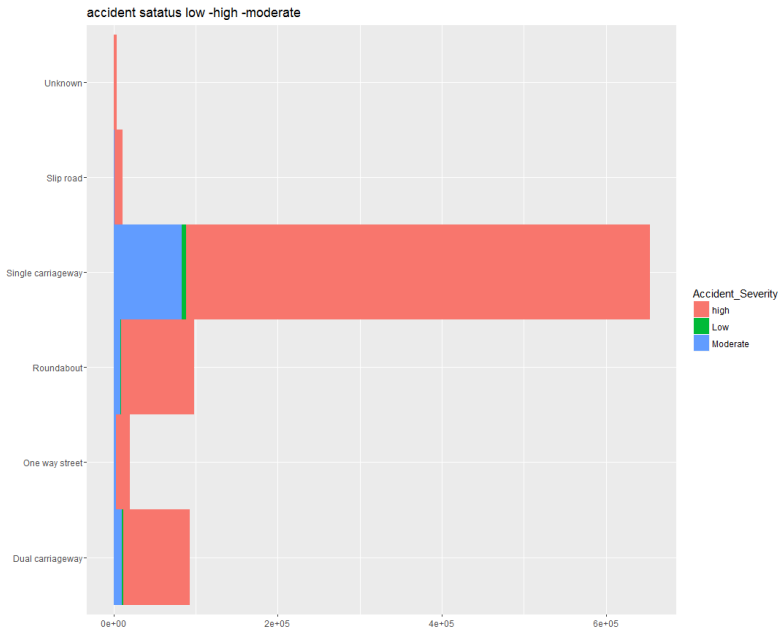


FIGURE7 60:ACCIDENT STATUES LOW -HIGH -MODERATE

bar + coord_polar()

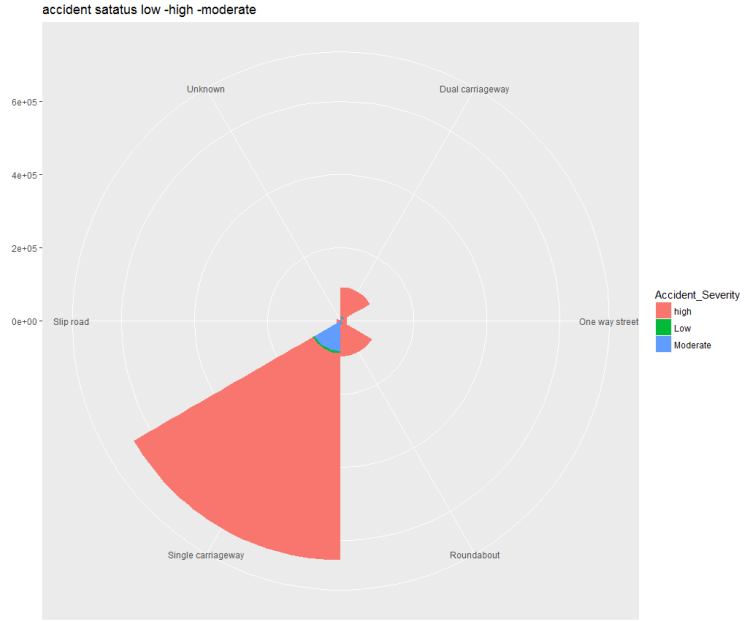


FIGURE7 61: ACCIDENT STATUES LOW -HIGH -MODERATE

-This figure show that the highest Accident Severity at road type is "Single carriageway"

7.2.4.13 the relation between road and light condtions.

```
# relation between raod and light condtions;
ggplot(data = accidents) +
 geom_count(mapping = aes(x =accidents$Road_Type ,y =accidents$Light_Conditions,color=..n..))+
 labs(
```

```
 title="increasing the number of accidents on single carriageway Roads ",
 subtitle="increasing accidents in cases of Daylight and darkeness when street lights is lit",
 x="Road type",
 y="light conditions",

 colour="accidents"


)+theme_bw()
```

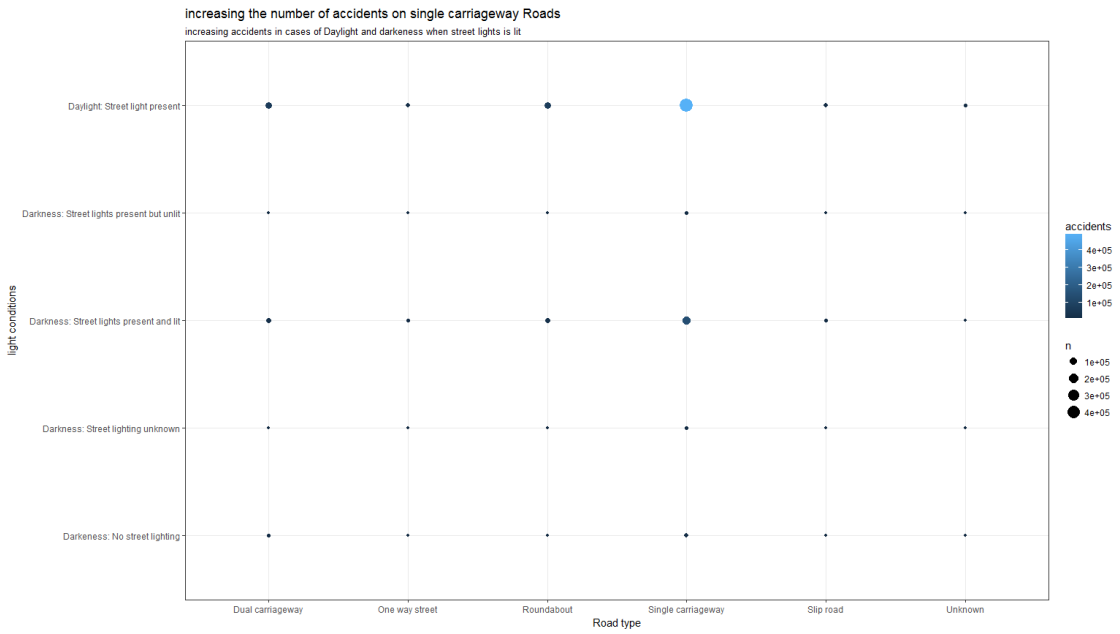


FIGURE7 62:INCREASING THE NUMBER OF ACCIDENTS ON SINGLE CARRIAGEWAY ROADS

-This figure show that the highest number of accidents at road type is "Single carriageway" and light condition is "Daylight: Street light present"

7.2.4.14 The number of accidents on each Road according to day of week.

```
#Number of accidents on each Road according to day of week

#plot:
ggplot(data = accidents, aes(x =Day_of_Week,fill=Day_of_Week)) +
 geom_bar()+

 labs(

  title="  Number of accidents on each Road according to day of week",
  subtitle="for All Days. ",
```

```
x="Days",
y="total number of accidents"


)+theme_bw()+facet_wrap(~Road_Type ,nrow=2)
```

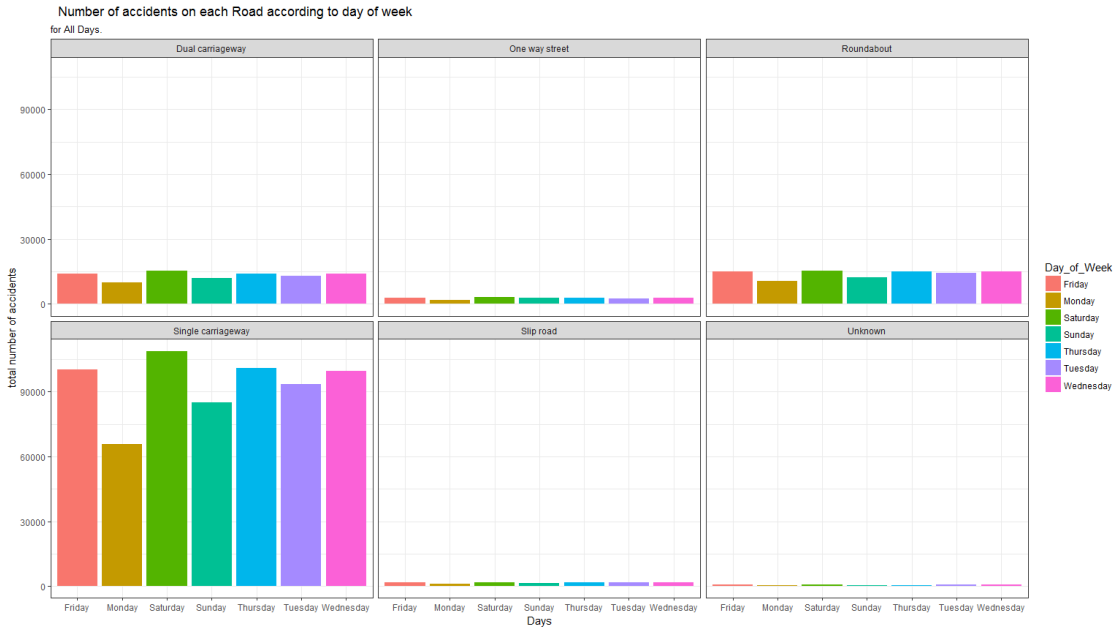


FIGURE7 63: NUMBER OF ACCIDENTS ON EACH ROAD ACCORDING TO DAY OF WEEK

-This figure show that the highest number of accidents at road type is "Single carriageway" at day of week is "Saturday"

7.2.4.15 The Number of accidents on each Road according to day of weekjunction controal condtions"

```
ggplot(data = accidents, aes(x = str_sub(Day_of_Week,1,3),fill=Day_of_Week)) +
 geom_bar()+

 labs(

  title="  Number of accidents on each Road according to day of week\nwith junction controal condtions",
  subtitle="for All Days. ",
  x="Days",
  y="total number of accidents"


)+theme_bw()+facet_grid(Junction_Control~Road_Type )
```

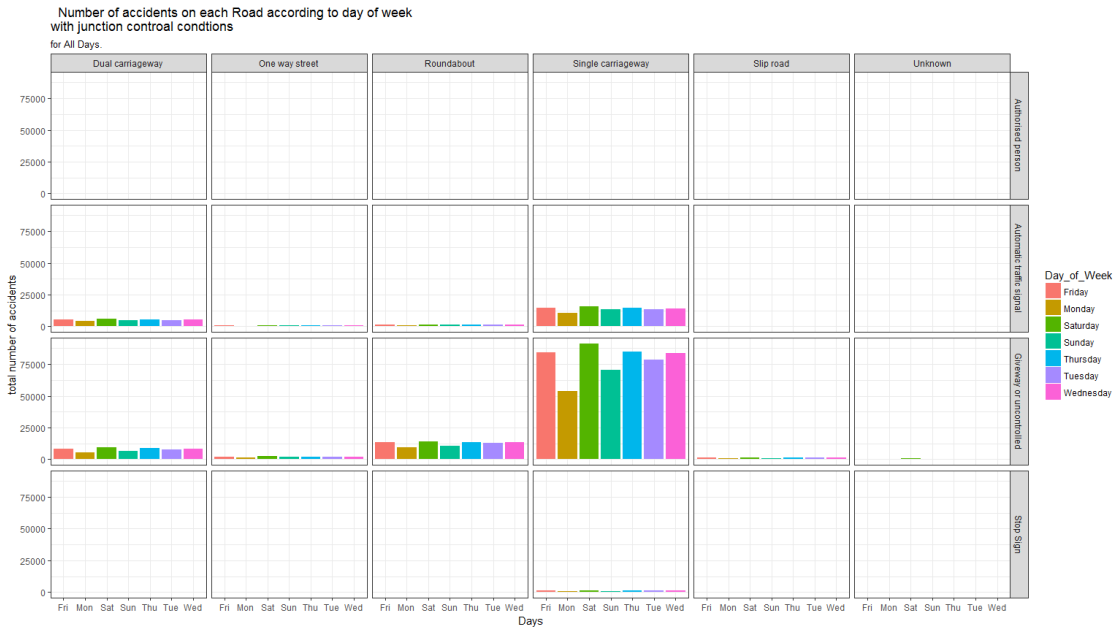


FIGURE7 64:NUMBER OF ACCIDENTS ON EACH ROAD ACCORDING TO DAY OF WEEK\NWITH JUNCTION CONTROAL CONDTIONS

-This figure show that the highest number of accidents at day of week is "Saturday" and road type is "Single carriageway", junction control is "Giveway or uncontrolled"

7.2.5 The fifth goal "what are the busiest roads in the nation? " .

7.2.5 .1 the relation between Region and the traffic volume on each Road.

```
ggplot(ukTraffic, aes(Region, RoadCategory)) +
 geom_jitter(aes(color = trafficvolume1), size = 0.5)+

 labs(

  subtittle="relation betwwen Region and the traffic jam in each Road",
  title="Busiest roads in the nation",
  x="Region",
  y="Roads",
  colour="traffic volume",
  caption="traffic Data"

 )
```

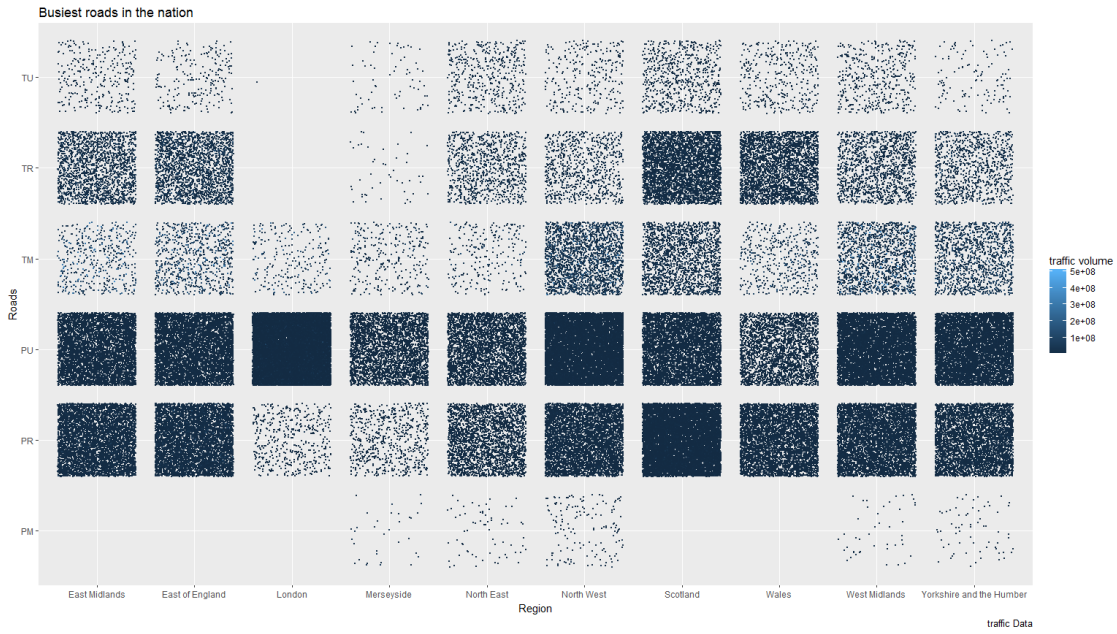


FIGURE7 65:BUSIEST ROADS IN THE NATION

-This figure show that the highest busiest roads in the region

7.2.5 .2 the relation between Annual Traffic volume by Road in each region in Britain.

Group traffic volume by year and region.

```
# groub traffic volume by year and region

s<-group_by(ukTraffic,AADFYear,Region,RoadCategory)
head(s)
```

```
## # A tibble: 6 x 29
## # Groups:   AADFYear, Region, RoadCategory [2]
##       X AADFYear    CP Estimation_method Region     LocalAuthority   Road
##   <int>    <int> <int> <fct>             <fct>      <fct>            <fct>
## 1     1     2005  6222 Estimated         Yorkshire~ North Lincolnsh~ A18
## 2     2     2005  6232 Counted           Yorkshire~ North Lincolnsh~ A18
## 3     3     2005  6233 Estimated         Yorkshire~ North Lincolnsh~ A18
## 4     4     2005  6234 Estimated         Yorkshire~ North East Linc~ A46
## 5     5     2005  6565 Estimated         Yorkshire~ Sheffield        A57
## 6     6     2005  6576 Estimated         Yorkshire~ Calderdale
```

Summarize to compute traffic volume in each road by year and region.

```
summary_trafic<-dplyr::summarize(s,total=sum(trafficvolume1),avearge=mean(trafficvolume1))
head(summary_trafic)
```

```
## # A tibble: 6 x 5
## # Groups:   AADFYear, Region [2]
```

```
##   AADFYear Region         RoadCategory     total   avearge
##     <int> <fct>          <fct>           <dbl>    <dbl>
## 1    2005 East Midlands   PR         5621187048  10506892
## 2    2005 East Midlands   PU         2674162813   4888780
## 3    2005 East Midlands   TM         4209791802 102677849
## 4    2005 East Midlands   TR         4143147766  23811194
## 5    2005 East Midlands   TU          336410390  12459644
## 6    2005 East of England PR         6840978029  12415568
```

Compute the rate of traffic flow according Road type.

```
summary_trafic<-mutate(summary_trafic,percent=total/sum(total)*100)

# show summarize:
head(summary_trafic)
```

```
## # A tibble: 6 x 6
## # Groups:   AADFYear, Region [2]
##   AADFYear Region         RoadCategory     total   avearge percent
##     <int> <fct>          <fct>           <dbl>    <dbl>   <dbl>
## 1    2005 East Midlands   PR         5621187048  10506892   33.1
## 2    2005 East Midlands   PU         2674162813   4888780   15.7
## 3    2005 East Midlands   TM         4209791802 102677849   24.8
## 4    2005 East Midlands   TR         4143147766  23811194   24.4
## 5    2005 East Midlands   TU          336410390  12459644    1.98
## 6    2005 East of England PR         6840978029  12415568   32.1
```

Plot: the relation between year and Annual traffic Volume.

```
#plot:
ggplot(data=summary_trafic,aes(x=AADFYear,y=avearge,color=RoadCategory))+geom_smooth(method="lo
ess",span=0.6,se=F)+ scale_x_discrete(limits = seq(2005,2016,by=4))+

 labs(

  title="Busiest roads in the nation\nAnnual Trafic volume by Road in each region in Britain",
  subtitle="Discovering Buisest Roads in  Britain",
  x="year",
  y="Annual traffic Volume(vichiles per miles)",
  color="Road"
 )+theme_bw()+facet_grid(RoadCategory~Region)
```

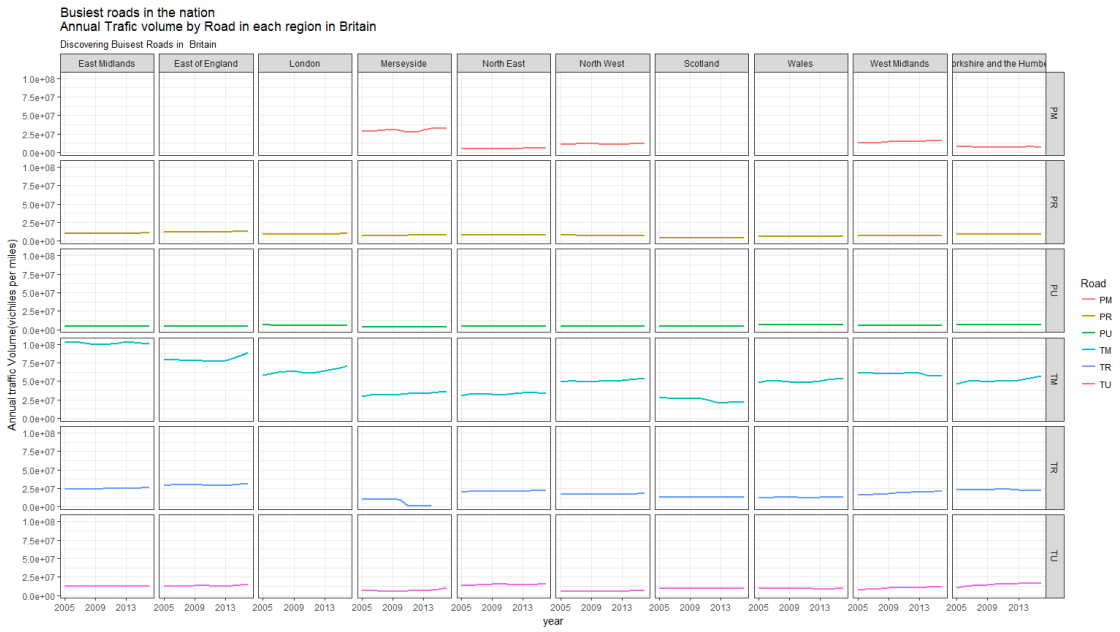


FIGURE7 66:BUSIEST ROADS IN THE NATION\NANNUAL TRAFIC VOLUME BY ROAD IN EACH REGION IN BRITAIN

-This figure show the highest busiest roads in each region upon their annual traffic volume

7.2.5 .3The Rate of traffic Volume on each Road.

Group traffic volume by RoadCategory.

```
# groub traffic volume by year and region

bi<-group_by(ukTraffic,RoadCategory)
head(bi)

## # A tibble: 6 x 29
## # Groups:   RoadCategory [2]
##       X AADFYear    CP Estimation_method Region     LocalAuthority   Road
##   <int>    <int> <int> <fct>             <fct>      <fct>            <fct>
## 1     1     2005  6222 Estimated         Yorkshire~ North Lincolnsh~ A18
## 2     2     2005  6232 Counted           Yorkshire~ North Lincolnsh~ A18
## 3     3     2005  6233 Estimated         Yorkshire~ North Lincolnsh~ A18
## 4     4     2005  6234 Estimated         Yorkshire~ North East Linc~ A46
## 5     5     2005  6565 Estimated         Yorkshire~ Sheffield
```

Summarize to compute traffic volume on each road.

```
buisetroad<-dplyr::summarize(bi,total=sum(trafficvolume1),avearge=mean(trafficvolume1))
head(summary_trafic)

## # A tibble: 6 x 6
## # Groups:   AADFYear, Region [2]
##   AADFYear Region         RoadCategory     total   avearge percent
```

```
##    <int> <fct>         <fct>          <dbl>    <dbl>   <dbl>
## 1    2005 East Midlands   PR         5621187048  10506892   33.1
## 2    2005 East Midlands   PU         2674162813   4888780   15.7
## 3    2005 East Midlands   TM          4209791802 102677849   24.8
## 4    2005 East Midlands   TR         4143147766  23811194   24.4
## 5    2005 East Midlands   TU          336410390  12459644    1.98
## 6    2005 East of England PR         6840978029  12415568   32.1
```

Compute the rate of traffic flow according Rod category.

```
buisetroad<-mutate(buisetroad,percent=total/sum(s$trafficvolume1)*100)

# show summarize:
head(buisetroad)
```

```
## # A tibble: 6 x 4
##   RoadCategory       total  avearge percent
##   <fct>              <dbl>    <dbl>   <dbl>
## 1 PM           4254114329 11816984   0.263
## 2 PR         394303786224  8016260  24.3
## 3 PU         417333496440  5429576  25.8
## 4 TM         474522386290 50985536  29.3
## 5 TR         298342113619 18613808  18.4
## 6 TU          31413124126 10746878   1.94
```

Plot:the relation between year Annual traffic Volume according Road catgory.

```
# all  Roads traffic volume rate



#plot:

bar <- ggplot(data = buisetroad,mapping = aes(x = RoadCategory, y = total,fill=RoadCategory)) +
  geom_bar( stat="identity", width = 1) +
  theme(aspect.ratio = 1) +
  labs(x = NULL, y = NULL,
          title="Rate of traffic Volume on each Road:"
         )

 bar + coord_flip()
```

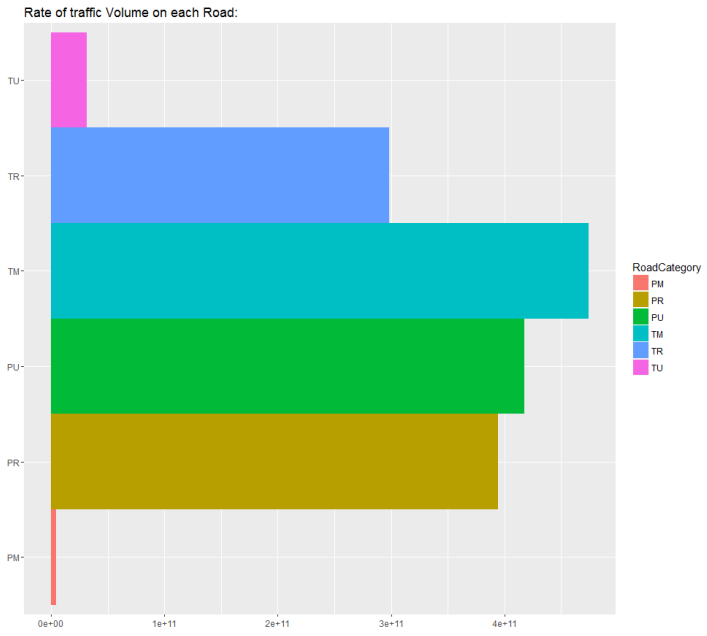


FIGURE7 67:RATE OF TRAFFIC VOLUME ON EACH ROAD

-This figure show the rate of traffic volume on each road category

bar + coord_polar()+geom_text(aes(label=paste0(round(percent,1),"%")),vjust=-1.6,size=4,color="gray9",fontface="bold")

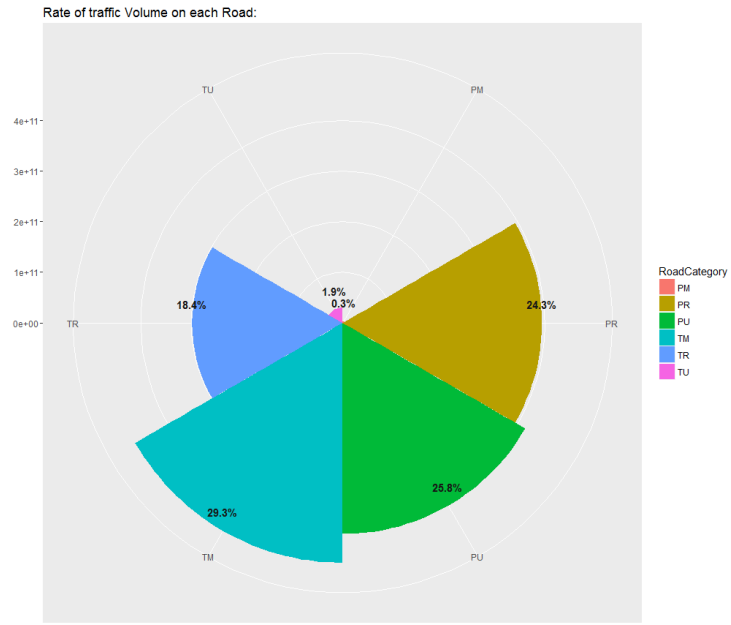


FIGURE7 68:RATE OF TRAFFIC VOLUME ON EACH ROAD

7.2.5 .3.1 Annual Trafic volume by yeras in Britain according Road category.

Group traffic volume by year and RoadCategory.

 # groub traffic volume by year and region

```
ro<-group_by(ukTraffic,AADFYear,RoadCategory)
head(ro)
```

```
## # A tibble: 6 x 29
## # Groups:   AADFYear, RoadCategory [2]
##       X AADFYear    CP Estimation_method Region     LocalAuthority   Road
##   <int>    <int> <int> <fct>             <fct>      <fct>            <fct>
## 1     1     2005  6222 Estimated         Yorkshire~ North Lincolnsh~ A18
## 2     2     2005  6232 Counted           Yorkshire~ North Lincolnsh~ A18
## 3     3     2005  6233 Estimated         Yorkshire~ North Lincolnsh~ A18
## 4     4     2005  6234 Estimated         Yorkshire~ North East Linc~ A46
## 5     5     2005  6565 Estimated         Yorkshire~ Sheffield        A57
## 6     6     2005  6576 Estimated         Yorkshire~ Calderdale       A58
## # ... with 22 more variables: RoadCategory <fct>, Easting <int>,
## #   Northing <int>, LinkLength_km <dbl>, LinkLength_miles <dbl>,
## #   PedalCycles <int>, Motorcycles <int>, CarsTaxis <int>,
## #   BusesCoaches <int>, LightGoodsVehicles <int>, V2AxleRigidHGV <int>,
## #   V3AxleRigidHGV <int>, V4or5AxleRigidHGV <int>,
## #   V3or4AxleArticHGV <int>, V5AxleArticHGV <int>,
## #   V6orMoreAxleArticHGV <int>, AllHGVs <int>, AllMotorVehicles <int>,
## #   Latitude <dbl>, Longitude <dbl>, trafficvolume <dbl>,
## #   trafficvolume1 <dbl>
```

Summarize to compute traffic volume in each road by year and Road category.

```
roadstraffic<-dplyr::summarize(ro,total=sum(trafficvolume1),aveargе=mean(trafficvolume1))
head(roadstraffic)
```

```
## # A tibble: 6 x 4
## # Groups:   AADFYear [1]
##   AADFYear RoadCategory      total  aveargе
##      <int> <fct>             <dbl>    <dbl>
## 1     2005 PM           374495701 11348355
## 2     2005 PR           34681142114  7956215
## 3     2005 PU           38666169723  5549902
## 4     2005 TM           40180896330 51846318
## 5     2005 TR           27266843295 17879897
## 6     2005 TU            2882081986  9972602
```

Compute the rate of traffic flow according Rod type.

```
roadstraffic<-mutate(roadstraffic,percent=total/sum(s$trafficvolume1)*100)

# show summarize:
head(roadstraffic)
```

```
## # A tibble: 6 x 5
## # Groups:   AADFYear [1]
##   AADFYear RoadCategory      total  aveargе percent
```

```
##    <int> <fct>          <dbl>    <dbl>   <dbl>
## 1   2005 PM         374495701 11348355  0.0231
## 2   2005 PR       34681142114  7956215  2.14
## 3   2005 PU       38666169723  5549902  2.39
## 4   2005 TM       40180896330 51846318  2.48
## 5   2005 TR       27266843295 17879897  1.68
## 6   2005 TU        2882081986  9972602  0.178
```

Plot: the relation between year Annual traffic Volume according Road catgory.

#1.2 buisest road:

#---------plot:

```
ggplot(data=roadstraffic,aes(x=AADFYear,y=total,color=RoadCategory))+
 geom_point()+ scale_x_discrete(limits = c(2005:2016))+geom_smooth(method = 'loess',se=F)+

 labs(

  title="Annual Trafic volume by yeras in Britain",
  subtitle="increasing traffic volume through years",
  x="year",
  y="Annual traffic Volume(vichiles per miles)"
)+theme_classic()+ geom_text(aes(label=paste0(round(percent,1),"%") ),vjust=-0.7,size=3)
```

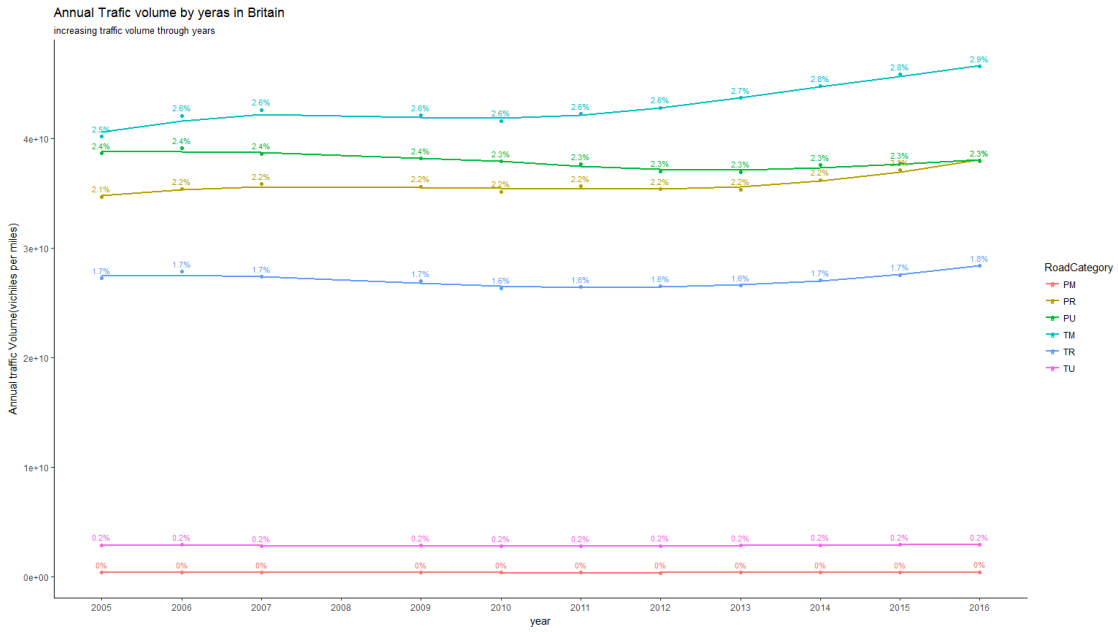


FIGURE7 69:ANNUAL TRAFIC VOLUME BY YERAS IN BRITAIN

7.2.6 Sixth goal "How has London has changed for cyclists? " .

In this goal want to discover if pedal cycles effect on the traffic volume in London from which the increasing or decreasing.

To achieve this goal should do the following:

1- Fisrt select the Data of London Region.

A-by Local authority in London.

- **Compute traffic volume in each local authority in London without pedal cylists.**
- **Compute traffic volume in each local authority in London with pedal cylists.**
- **Compare grapgh and observe the change in grpgh if pedal cyles effect on traffic volume in each local authority or not.**

B- Change pedal cyclists on each Road in London.

- **Compute traffic volume for pedal cyclists only on each Road category.**
- **Compute traffic volume for car types on each Road without pedal cylists.**
- **compute traffic volume for car types on each Road with pedal cylists.**
- **Compare the graph and note if pedal cylists effect on the traffic flow on each Road or not.**

c- Change pedal cylists through years.

'

7.2.6.1 Select data of London only.

```
london<-filter(ukTraffic,ukTraffic$Region=="London")
```

By Local autority in London.

7.2.6.1.1 Each LocalAuthority in London and traffic volume in each Road without pedal cycles.

Plot: without pedal cycles.

```
#plot:
ggplot(data = london, mapping = aes(x =reorder(LocalAuthority,trafficvolume,FUN=median), y =
london$trafficvolume,fill=london$RoadCategory)) +
 geom_boxplot()+
 labs(

  title="the Busiest roads in london with cars\nwithout pedal cycles: ",
  x="Local Authority",
  y="Traffic volume",
  fill="Road category"

 )+coord_flip()
```

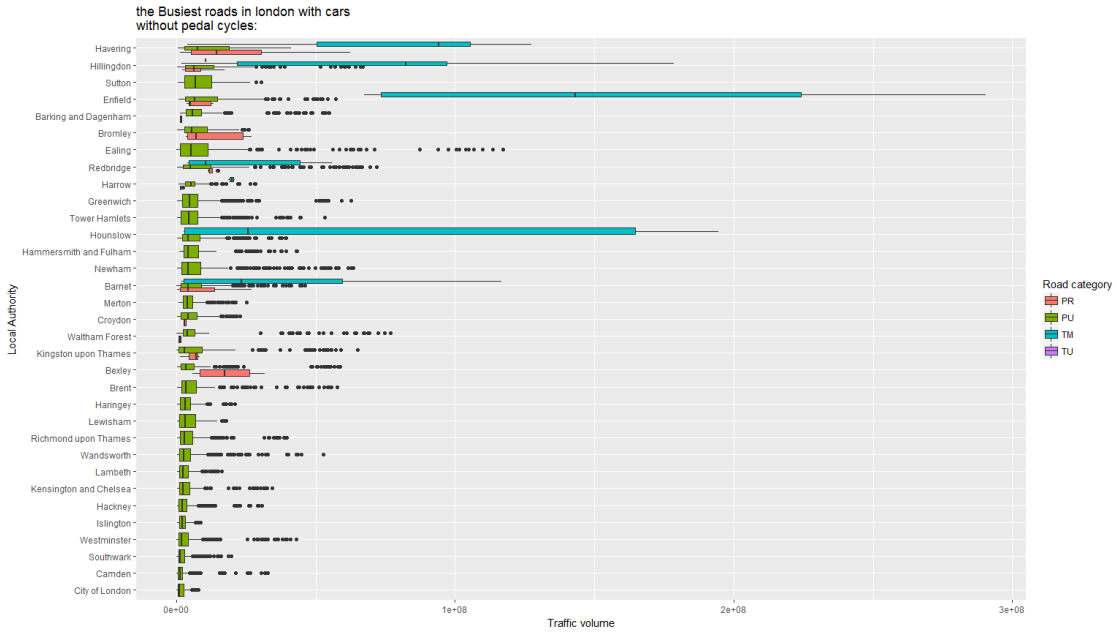


FIGURE7 70:THE BUSIEST ROADS IN LONDON WITH CARS\NWITHOUT PEDAL CYCLES

-This figure show the median of traffic volume at local authority in London and traffic volume for every road category this traffic volume without pedal cycle

7.2.6.1.2 Each local Authority in London and traffic volume in each Road With pedal cycles: only

```
# to convert AADF to traffic volume.
london<-mutate(london,pedalvolume=(london$PedalCycles*london$LinkLength_miles)*365)

# #1-each local Authority in London and traffic volume in each Road.
# with pedal cycles:only
ggplot(data = london, mapping = aes(x =reorder(london$LocalAuthority,london$pedalvolume,FUN=median),
y = london$PedalCycles,fill=london$RoadCategory)) +
 geom_boxplot()+
 labs(

  title="london  and the Busiest roads  with pedalcyles.",
  x="Local Authority",
  y="motorcycles",
  fill="Road category"

 )+coord_flip()
```

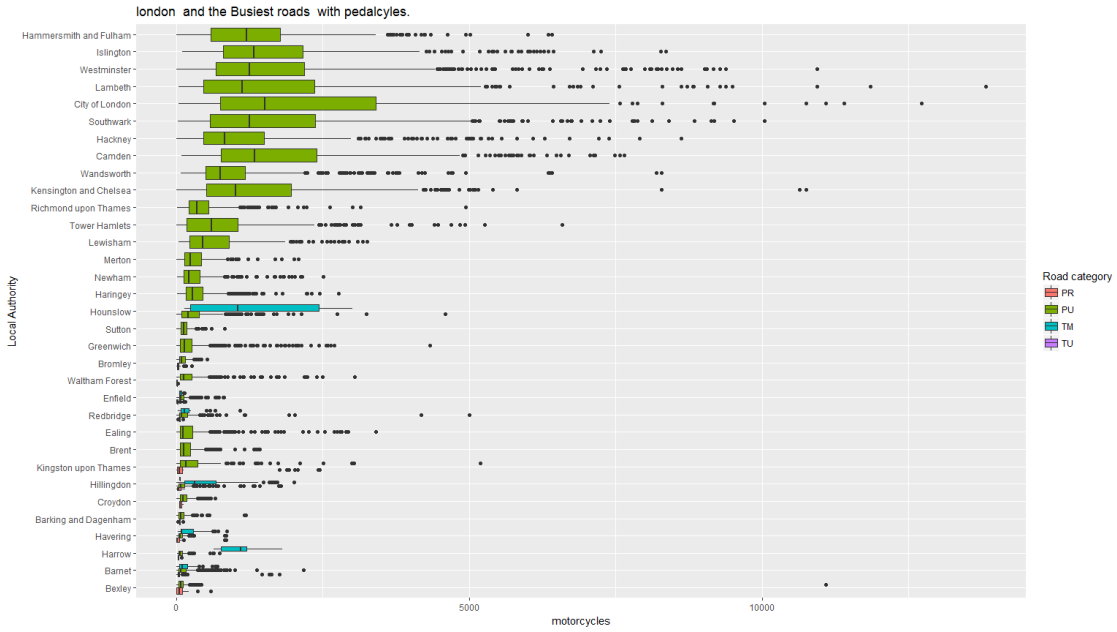


FIGURE7 71: LONDON AND THE BUSIEST ROADS WITH PEDALCYLES

-This figure show the median traffic volume of pedal cycle only which we can get by multiply pedal cycles , link length miles and 356 so it show traffic volume of pedal cycle at local authority in London by road category

B- Change pedal cyclists on each Road in London.

7.2.6.2 Compute traffic volume For cars without pedal cycles on each Road Category.

steps:

Summarize by Road category:

```
# summarize by Road category:
csr<-group_by(london,RoadCategory)
allcartraffic<-dplyr::summarize(csr,volume=sum(trafficvolume),mean=mean(trafficvolume))
head(allcartraffic)
```

```
## # A tibble: 4 x 3
##   RoadCategory      volume     mean
##   <fct>              <dbl>    <dbl>
## 1 PR           4454732520  9642278
## 2 PU         111879899718  5757508
## 3 TM          15363976635 63487507
## 4 TU             10448271 10448271
```

Compute the Rate of traffic volume on each Road.

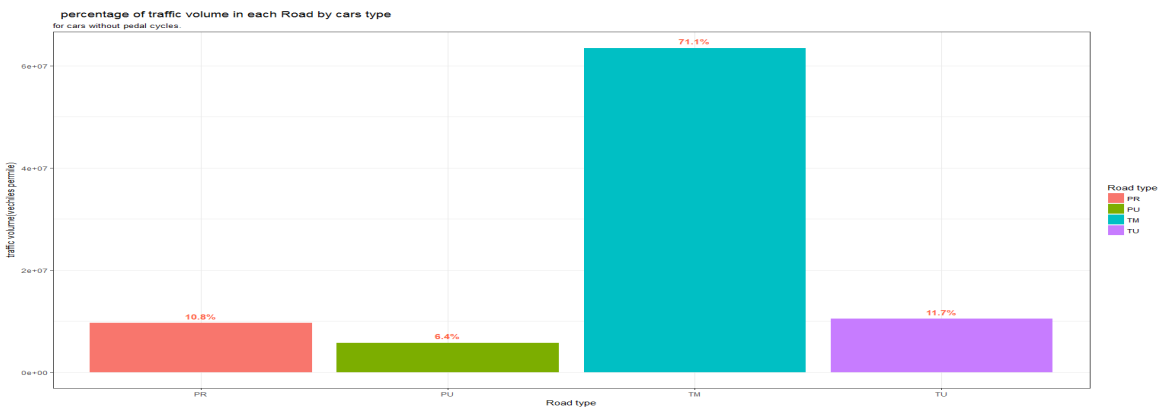
```
# compute percentage %
allcartraffic<-mutate(allcartraffic,percent=paste0(round(mean/sum(allcartraffic$mean)*100,1),"%"))

head(allcartraffic)
```

```
## # A tibble: 4 x 4
##   RoadCategory      volume     mean percent
##   <fct>              <dbl>    <dbl> <chr>
## 1 PR           4454732520  9642278 10.8%
## 2 PU         111879899718  5757508 6.4%
## 3 TM          15363976635 63487507 71.1%
## 4 TU             10448271 10448271 11.7%
```

Plot: The relation between traffic volume and Road type without pedal cylcles.

```
#plot
```




```
ggplot(data=allcartraffic,
```

```
aes(x = RoadCategory,y=allcartraffic$mean, fill =allcartraffic$RoadCategory)) +
 geom_bar(stat="identity")+

 labs(

  title="  percentage of traffic volume in each Road by cars type ",
  subtitle="for cars without pedal cycles. ",
  x="Road type",
  y="traffic volume(vechiles permile)",
  fill="Road type"

 )+theme_bw()+geom_text(aes(label=percent),vjust=-0.7,size=4,fontface="bold",color="tomato1")
```

FIGURE7 72:PERCENTAGE OF TRAFFIC VOLUME IN EACH ROAD BY CARS TYPE

This figure show the percentage of traffic volume of cars on each road type and we can find that the highest percentage of traffic volume of cars

7.2.6.2.1 Second compute traffic volume for pedal cyles on each Road:

steps:

Summarize by Road category:

```
csr1<-group_by(london,RoadCategory)

pedal<-dplyr::summarize(csr1,volume=sum(pedalvolume),mean=mean(pedalvolume))
head(pedal)
```

```
## # A tibble: 4 x 3
##   RoadCategory    volume   mean
##   <fct>            <dbl>  <dbl>
## 1 PR           11866997  25686
## 2 PU         2129322137 109578
```

```
## 3 TM          36859799 152313
## 4 TU            17148  17148
```

Compute the Rate of traffic volume on each Road in addition to pedal cycles.

```
#compute percentage %
pedal<-mutate(pedal,percent=paste0(round(mean/sum(pedal$mean)*100,1),"%"))

head(pedal)
```

```
## # A tibble: 4 x 4
##   RoadCategory    volume   mean percent
##   <fct>            <dbl>  <dbl> <chr>
## 1 PR             11866997  25686 8.4%
## 2 PU           2129322137 109578 36%
## 3 TM             36859799 152313 50%
## 4 TU               17148  17148 5.6%
```

Plot: The relation between traffic volume and Road type with pedal cylcles only.

```
# second compue traffic volume for pedal cyles only  on each Road :




#plot:

ggplot(data = pedal, aes(x = RoadCategory,y=pedal$mean, fill =pedal$RoadCategory)) +
 geom_bar(stat="identity")+

 labs(

  title="  percentage of traffic volume in each Road by pedal cyles only ",
  subtitle="for pedal cycles. ",
  x="Road type",
  y="traffic volume(vechiles per mile)",
  fill="Road type"

 )+theme_bw()+geom_text(aes(label=percent),vjust=-0.7,size=4,fontface="bold",color="tomato1")
```

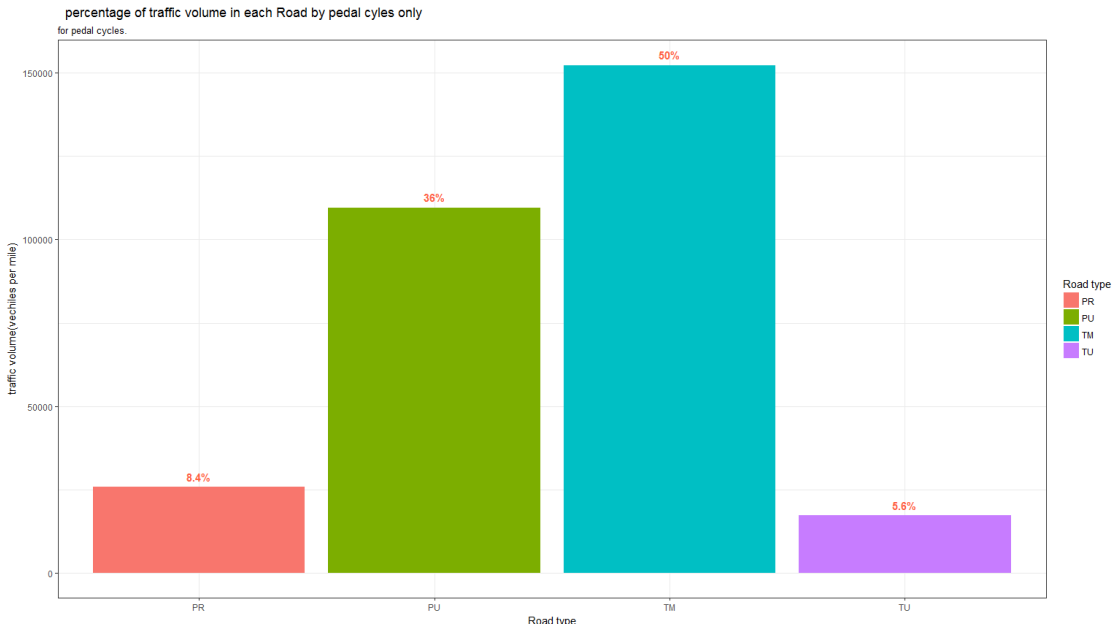


FIGURE7 73:PERCENTAGE OF TRAFFIC VOLUME IN EACH ROAD BY PEDAL CYLES ONLY

-This figure show the percentage of traffic volume of pedal cycle on each road type and we can find that the highest percentage of traffic volume of pedal cycle on road type "TM"

C- Change pedal cyclists through years.

1-compue traffic volume for pedal cyles only by years: to show change:

steps:

Summarize by year to compute traffic volume.

```
csr2<-group_by(london,AADFYear)

pedal2<-dplyr::summarize(csr2,volume=sum(pedalvolume),mean=mean(pedalvolume))

head(pedal2)
```

```
## # A tibble: 6 x 3
##   AADFYear    volume   mean
##      <int>     <dbl>  <dbl>
## 1     2005 166931469  91070
## 2     2006 196409712 107094
## 3     2007 181296274  98691
## 4     2009 183926205 100123
## 5     2010 191427849 104264
## 6     2011 200726016 109987
```

Compute the Rate of traffic volume in each year for pedal cycles only.

```
# compute percentage %
pedal2<-mutate(pedal2,percent=paste0(round(mean/sum(pedal2$mean)*100,1),"%"))

head(pedal2)
```

```
## # A tibble: 6 x 4
##   AADFYear    volume   mean percent
##      <int>     <dbl>  <dbl> <chr>
## 1     2005 166931469  91070 7.7%
## 2     2006 196409712 107094 9%
## 3     2007 181296274  98691 8.3%
## 4     2009 183926205 100123 8.4%
## 5     2010 191427849 104264 8.8%
## 6     2011 200726016 109987 9.2%
```

Plot: the relation between Year and traffic volume.

```
# pedal cyles by year:

# second compue traffic volume for pedal cycles only  by years:




#plot

ggplot(data = pedal2, aes(x = AADFYear,y=pedal2$volume)) +
 geom_bar(stat="identity",fill="DarkOliveGreen")+

 labs(

  title="  percentage of traffic volume for pedal cycles  through years",
  subtitle="for pedal cycles. ",
  x="Year",
  y="traffic volume(vehicles per mile)"


 )+theme_bw()+geom_text(aes(label=percent),vjust=-
0.7,size=4,fontface="bold",color="DarkCyan")+scale_x_discrete(limits=seq(2005,2016,by=1))
```

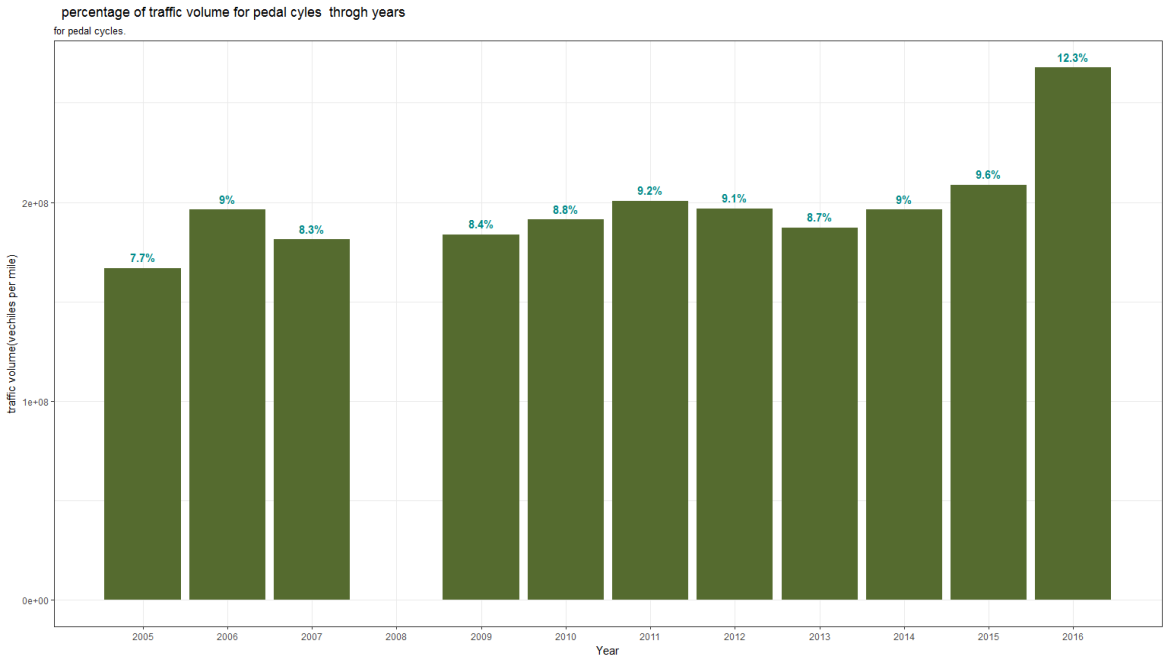


FIGURE7 74:PERCENTAGE OF TRAFFIC VOLUME FOR PEDAL CYCLES THROUGH YEARS

-This figure show the percent for every traffic volume of pedal cycle through years and we can note that this percent is increasing

7.2.6.2.2 Compute traffic volume for all cars types through years:

Steps:

Summarize by year to compute traffic volume.

```
csr3<-group_by(london,AADFYear)

allcrs<-dplyr::summarize(csr3,volume=sum(trafficvolume),mean=mean(trafficvolume))
head(allcrs)
```

```
## # A tibble: 6 x 3
##   AADFYear     volume    mean
##      <int>      <dbl>   <dbl>
## 1     2005 12422281308 6777022
## 2     2006 12709770945 6930082
## 3     2007 12416589553 6759167
## 4     2009 12076519341 6574044
## 5     2010 12068054528 6573014
## 6     2011 11677685371 6398732
```

Compute the Rate of traffic volume in each year for car types only.

```
# compute percentage %
allcrs<-mutate(allcrs,percent=paste0(round(mean/sum(allcrs$mean)*100,2),"%"))
head(allcrs)
```

```
## # A tibble: 6 x 4
##   AADFYear     volume    mean percent
##      <int>      <dbl>   <dbl> <chr>
## 1     2005 12422281308 6777022 9.42%
## 2     2006 12709770945 6930082 9.63%
## 3     2007 12416589553 6759167 9.4%
## 4     2009 12076519341 6574044 9.14%
## 5     2010 12068054528 6573014 9.14%
## 6     2011 11677685371 6398732 8.89%
```

Plot: the relation between Year and traffic volume for car types.

```
#traffic for all cars through years:



#plot

ggplot(data = allcrs, aes(x = AADFYear,y=allcrs$volume)) +
 geom_bar(stat="identity",fill="DarkCyan")+

 labs(

  title="  percentage of traffic volume for cars  through years",
  subtitle="for  ALL cars in London. ",
  x="Year",
  y="traffic volume(vehicles per mile)"


 )+theme_bw()+geom_text(aes(label=percent),vjust=-
0.7,size=4,fontface="bold",color="DarkOliveGreen")+scale_x_discrete(limits=seq(2005,2016,by=1))
```

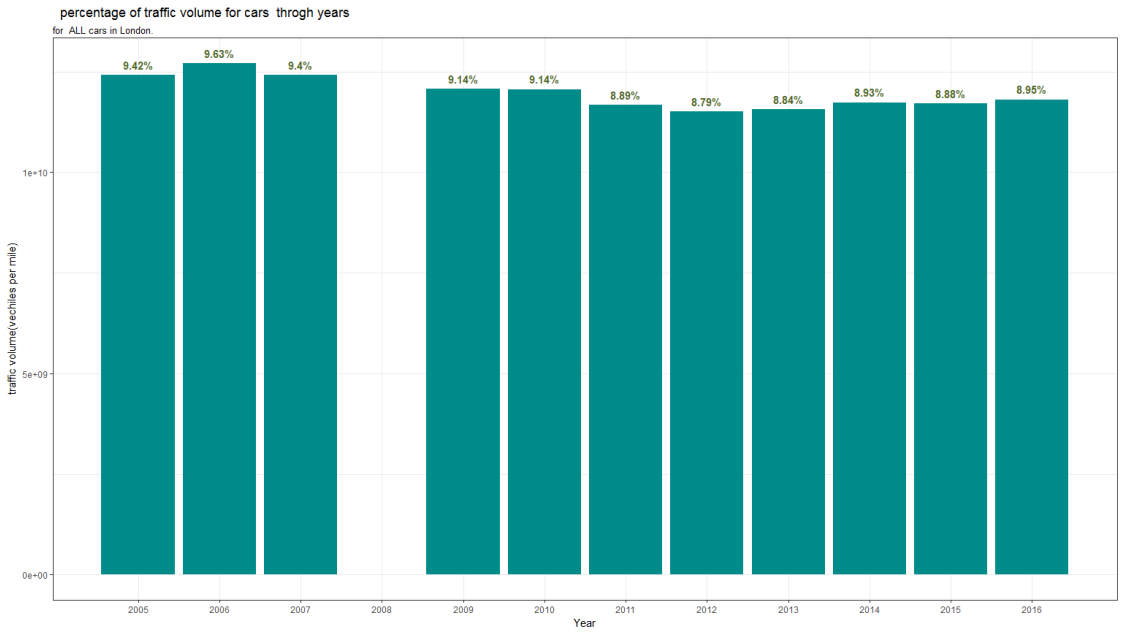


FIGURE7 75:PERCENTAGE OF TRAFFIC VOLUME FOR CARS THROUGH YEARS

-This figure show the percent for every traffic volume of cars through years and we can note that this percent is decreasing as people use pedal cycle

7.2.7 The seventh Goal " Which areas changed and never change? and why ? " .

To acieve this Goal shoud do the follwing.

7.2.7.1 The first compute the traffic volume rate in each area and determine The most crowded areas statistically.

Steps:

Group the traffic volume By Region.

```
change2<-group_by(ukTraffic,Region)
```

Summarizing the traffic volume according to each Region.

```
areachange2<-dplyr::summarize(change2,volume=sum(trafficvolume1),mean=mean(trafficvolume1))
head(areachange2)
```

```
## # A tibble: 6 x 3
##   Region           volume     mean
##   <fct>             <dbl>    <dbl>
## 1 East Midlands   189133580146 13001552
## 2 East of England 240803923433 15063426
```

```
## 3 London         133887123224  6648812
## 4 Merseyside      31048918709  6349472
## 5 North East      78456099745  9112207
## 6 North West      228345768122 11295299
```

Compute the rate of traffic volume in each Region.

```
areachange2<-mutate(areachange2,percent=paste0(round(mean/sum(areachange2$mean)*100,2),"%"))

head(areachange2)
```

```
## # A tibble: 6 x 4
##   Region             volume     mean percent
##   <fct>               <dbl>    <dbl> <chr>
## 1 East Midlands   189133580146 13001552 12.56%
## 2 East of England 240803923433 15063426 14.55%
## 3 London         133887123224  6648812 6.42%
## 4 Merseyside      31048918709  6349472 6.13%
## 5 North East      78456099745  9112207 8.8%
## 6 North West      228345768122 11295299 10.91%
```

Plot: the gragh.

```
# eache area traffic volume Rate:


# plot


ggplot(data = areachange2, aes(x=Region,y=mean,fill=Region)) +
 geom_bar(stat="identity")+

 labs(

  title="  Which area traffic Volume change\nEache area traffic volume Rate:",
  subtitle="for All Regions. ",
  x="Region type",
  y="traffic volume(vechiles per mile)"


 )+theme_bw()+geom_text(aes(label=percent),vjust=-0.8,size=4,fontface="bold",color="DarkOliveGreen")
```

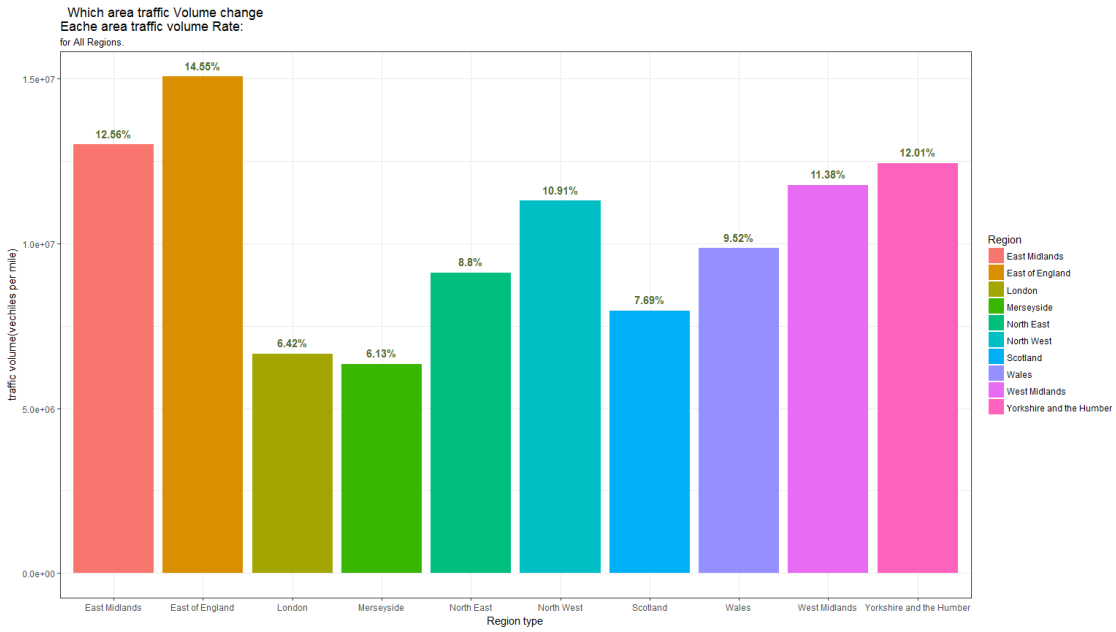


FIGURE7 76:WHICH AREA TRAFFIC VOLUME CHANGE\NEACHE AREA TRAFFIC VOLUME RATE

-This figure show the percentage of traffic volume for each region type upon their region and we can see that region type “East of England” the highest region of traffic volume

7.2.7.2 We study the change of traffic rate through years to determine which area change and not change.

Steps:

Groub by year and region to show change in traffic in each region throgh Years:

```
change<-group_by(ukTraffic,AADFYear,Region)
```

Summarize to compute mean:

```
areachange<-dplyr::summarize(change,volume=sum(trafficvolume1),mean=mean(trafficvolume1))

#show

head(areachange)
```

```
## # A tibble: 6 x 4
## # Groups:   AADFYear [1]
##   AADFYear Region            volume     mean
##      <int> <fct>              <dbl>    <dbl>
## 1     2005 East Midlands   16984699818 12828323
## 2     2005 East of England 21280392159 14726915
## 3     2005 London          12589212777  6868092
```

```
## 4    2005 Merseyside      2760804819  6148786
## 5    2005 North East      6924529376  8900423
## 6    2005 North West      20474761804 11200636
```

Compute percentage of traffic in each year:%.

```
areachange<-mutate(areachange,percent=paste0(round(mean/sum(areachange$mean)*100,1),"%"))
#show

head(areachange)
```

```
## # A tibble: 6 x 5
## # Groups:   AADFYear [1]
##   AADFYear Region            volume     mean percent
##      <int> <fct>              <dbl>    <dbl> <chr>
## 1    2005 East Midlands   16984699818 12828323 1.1%
## 2    2005 East of England 21280392159 14726915 1.3%
## 3    2005 London          12589212777  6868092 0.6%
## 4    2005 Merseyside       2760804819  6148786 0.5%
## 5    2005 North East       6924529376  8900423 0.8%
## 6    2005 North West      20474761804 11200636 1%
```

Plot: the relation between year and traffic volume according to each Region.

```
#plot

ggplot(data = areachange, aes(x = AADFYear,y=mean,color=Region)) +
 geom_point(size=5)+geom_smooth(se=F,method = 'gam')+

 labs(

  title=" which area never change?\n percentage of traffic volume  IN each Region",
  subtitle="for pedal cycles. ",
  x="Year",
  y="traffic volume(vechiles per mile)"

 )+theme_bw()+scale_x_discrete(limits=seq(2000,2016,by=1))+geom_text(aes(label=percent),vjust=-
0.8,size=4,fontface="bold",color="DarkOliveGreen")
```

FIGURE7 77:WHICH AREA NEVER CHANGE?\N PERCENTAGE OF TRAFFIC VOLUME IN EACH REGION

-This figure show the rate of traffic volume through years for every region and we can see that most region's rate is increasing except London which use pedal cycle so it's traffic volume is decreasing

7.2.7.3 determine the reasons why these areas the traffic change in it and why the traffic don't change in onther.tis acieved by study the traffic flow of cars in each region and determine which type lead to increasing and decreasing traffic flow in each region.

We divide the cars traffic flow into:

A- Cars type:

- **Motorcycles**
- **CarsTaxis,**
- **BusesCoaches,**
- **LightGoodsVehicles,**
- **Pedalcylist.**

We do the follwing on those types.

1-Convert Annual Average daily flow to Traffic volume.

```
# groub by Region:
regioncr<-group_by(ukTraffic,Region)

# summarize to compute total volume for each car
carstrffic<-dplyr::summarize(regioncr,

                  Motorcycles=mean(Motorcycles*LinkLength_miles*365),

                  CarsTaxis =mean(CarsTaxis*LinkLength_miles*365),

                  BusesCoaches =mean(BusesCoaches*LinkLength_miles*365) ,

                  LightGoodsVehicles=mean(LightGoodsVehicles*LinkLength_miles*365),
                  pedalcylist=mean(PedalCycles*LinkLength_miles*365))

head(carstrffic)
```

```
## # A tibble: 6 x 6
##   Region   Motorcycles CarsTaxis BusesCoaches LightGoodsVehic~ pedalcylist
##   <fct>        <dbl>    <dbl>       <dbl>          <dbl>      <dbl>
## 1 East Mi~     76737   9834000       68575        1720862      26641
## 2 East of~    103325  11510506       81949        2076829      43488
## 3 London      151775   5094436      143914         846508     108162
## 4 Merseys~     33629   5085293       71466         802996      17672
## 5 North E~     50338   7264153       72577        1199720      21944
## 6 North W~     56442   8813857       76495        1411375      29584
```

2- Make gather for traffic volume for car types: without V2AxleRigidHGV type to compute traffic volume in each Region according to car types.

```
carstrffic<-
gather(carstrffic,c(Motorcycles,CarsTaxis,BusesCoaches,LightGoodsVehicles,pedalcylist),key="carType",value="volume")

head(carstrffic)
```

```
## # A tibble: 6 x 3
##   Region        carType     volume
##   <fct>         <chr>       <dbl>
## 1 East Midlands   Motorcycles  76737
## 2 East of England Motorcycles 103325
## 3 London          Motorcycles 151775
## 4 Merseyside      Motorcycles  33629
## 5 North East      Motorcycles  50338
## 6 North West      Motorcycles  56442
```

3- Compute rate of traffic volume of each cars type in each Region.

```
# compue rate %

carstrffic<-mutate(carstrffic,percent=paste0(round(volume/sum(carstrffic$volume)*100,2),"%"))

head(carstrffic)
```

```
## # A tibble: 6 x 4
##   Region        carType     volume percent
##   <fct>         <chr>       <dbl> <chr>
## 1 East Midlands   Motorcycles  76737 0.08%
## 2 East of England Motorcycles 103325 0.11%
## 3 London          Motorcycles 151775 0.16%
## 4 Merseyside      Motorcycles  33629 0.04%
## 5 North East      Motorcycles  50338 0.05%
## 6 North West      Motorcycles  56442 0.06%
```

4- Plot the traffic volume of each Region according to Car types.

 # cars traffic traffic volume rate in each Region:

```
ggplot(data = carstrffic, aes(x =str_sub(carType,1,5) ,y=volume,fill= carType)) +
 geom_bar(stat="identity",position="dodge")+

 labs(

  title="  percentage of traffic volume for cars  through Region:",
  subtitle="for  ALL cars in   Britain. ",

  y="traffic volume(vechiles per mile)",
  x="Car Type"



 )+theme_bw()+geom_text(aes(label=percent),vjust=-
0.5,size=3,fontface="bold",color="Sienna")+facet_wrap(~Region,nrow=2)
```

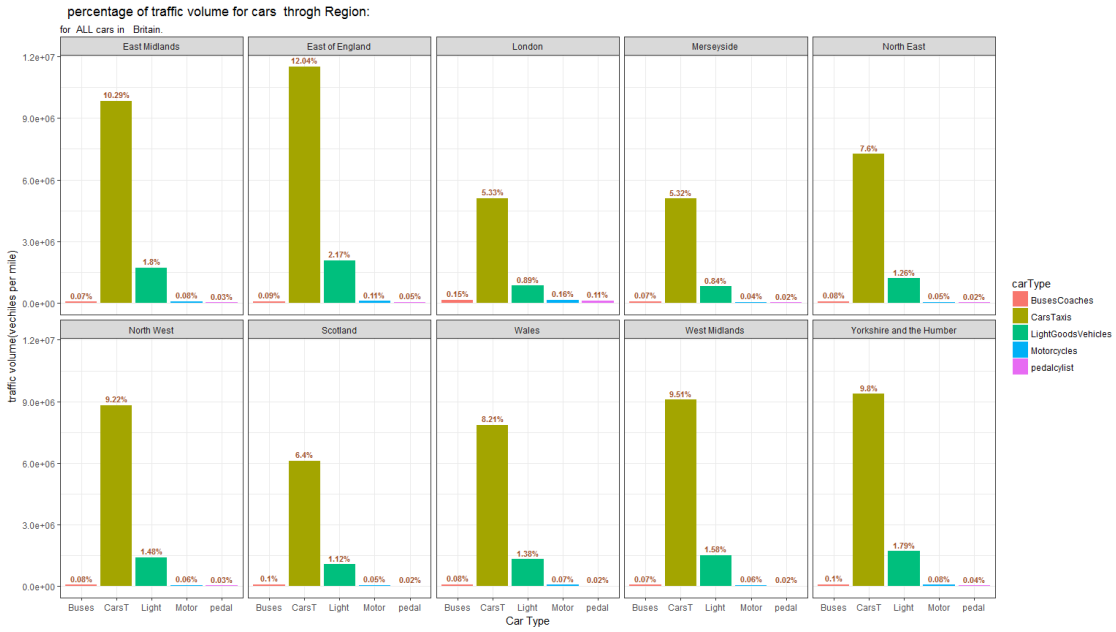


FIGURE7 78: PERCENTAGE OF TRAFFIC VOLUME FOR CARS THROUGH REGION

-This figure show the rate of traffic volume for every region upon their cars type and we can find that cars taxies are base for traffic flow in regions

B- Heavy Good vechiles (HGV):1-V2AxleRigidHGV ,2-V3AxleRigidHGV,3-V4or5AxleRigidHGV,4-V3or4AxleArticHGV,5-V4or5AxleRigidHGV.then do the follwing on that types.

1-Convert Annual Average daily flow to Traffic volume.

```
# groub by Region the V2AxleRigidHGV :
regioncr2<-group_by(ukTraffic,Region)

# summarize to compute total volume for  the V2AxleRigidHGV :
carstrffic2<-dplyr::summarize(regioncr2,


 V2AxleRigidHGV=mean(V2AxleRigidHGV*LinkLength_miles*365),

 V3AxleRigidHGV=mean(V3AxleRigidHGV*LinkLength_miles*365),

 V4or5AxleRigidHGV =mean(V4or5AxleRigidHGV*LinkLength_miles*365) ,


 V3or4AxleArticHGV=mean(V3or4AxleArticHGV*LinkLength_miles*365),
 V5AxleArticHGV=mean(V5AxleArticHGV*LinkLength_miles*365),

 V6orMoreAxleArticHGV=mean(V6orMoreAxleArticHGV*LinkLength_miles*365))

head(carstrffic2)
```

```
## # A tibble: 6 x 7
##   Region   V2AxleRigidHGV V3AxleRigidHGV V4or5AxleRigidH~ V3or4AxleArticH~
##   <fct>           <dbl>         <dbl>          <dbl>          <dbl>
## 1 East Mi~       344073         63517          69005          77440
## 2 East of~       375884         61925          71479          65653
## 3 London         156977         22464          33501           9396
## 4 Merseys~       119671         21167          26167          12496
## 5 North E~       172946         38568          43419          24526
## 6 North W~       287370         49655          52162          42720
## # ... with 2 more variables: V5AxleArticHGV <dbl>,
## #   V6orMoreAxleArticHGV <dbl>
```

2- Make gather for traffic volume for car types: without V2AxleRigidHGV type to compute traffic volume in each Region according to HGV.

```
#gather type of AxleRigidHGV:

carstrffic2<-
gather(carstrffic2,c(V2AxleRigidHGV,V3AxleRigidHGV,V4or5AxleRigidHGV,V3or4AxleArticHGV,V5AxleArticHGV,V6orMoreAxleArticHGV),key="allAxleRigidHGV",value="volume")

head(carstrffic2)
```

```
## # A tibble: 6 x 3
##   Region         allAxleRigidHGV volume
##   <fct>          <chr>           <dbl>
```

```
## 1 East Midlands   V2AxleRigidHGV  344073
## 2 East of England V2AxleRigidHGV  375884
## 3 London          V2AxleRigidHGV  156977
## 4 Merseyside      V2AxleRigidHGV  119671
## 5 North East      V2AxleRigidHGV  172946
## 6 North West      V2AxleRigidHGV  287370
```

3-compute rate of traffic volume of each HGV in each Region.

```
# compue rate %

carstrffic2<-mutate(carstrffic2,percent=paste0(round(volume/sum(carstrffic2$volume)*100,2),"%"))


# show sumarize:
head(carstrffic2)
```

```
## # A tibble: 6 x 4
##   Region         allAxleRigidHGV volume percent
##   <fct>          <chr>           <dbl> <chr>
## 1 East Midlands   V2AxleRigidHGV  344073 4.35%
## 2 East of England V2AxleRigidHGV  375884 4.75%
## 3 London          V2AxleRigidHGV  156977 1.98%
## 4 Merseyside      V2AxleRigidHGV  119671 1.51%
## 5 North East      V2AxleRigidHGV  172946 2.19%
## 6 North West      V2AxleRigidHGV  287370 3.63%
```

4-Plot the traffic volume of each Region according to HGV.

```
# second compue traffic for V2AxleRigidHGV types in aech Region:


# plot:

ggplot(data = carstrffic2, aes(x =str_sub(allAxleRigidHGV,1,5),y=volume,fill=allAxleRigidHGV )) +
 geom_bar(stat="identity",position="dodge")+

 labs(

  title="  percentage of traffic volume for allAxleRigidHGV  throgh Region:",
  subtitle="for  ALL allAxleRigidHGV in All Region . ",

  y="traffic volume(vechiles per mile)",

  x="allAxleRigidHGV type"


 )+theme_bw()+geom_text(aes(label=percent),vjust=-
0.4,size=3,fontface="bold",color="Sienna")+facet_wrap(~Region,nrow=2)
```

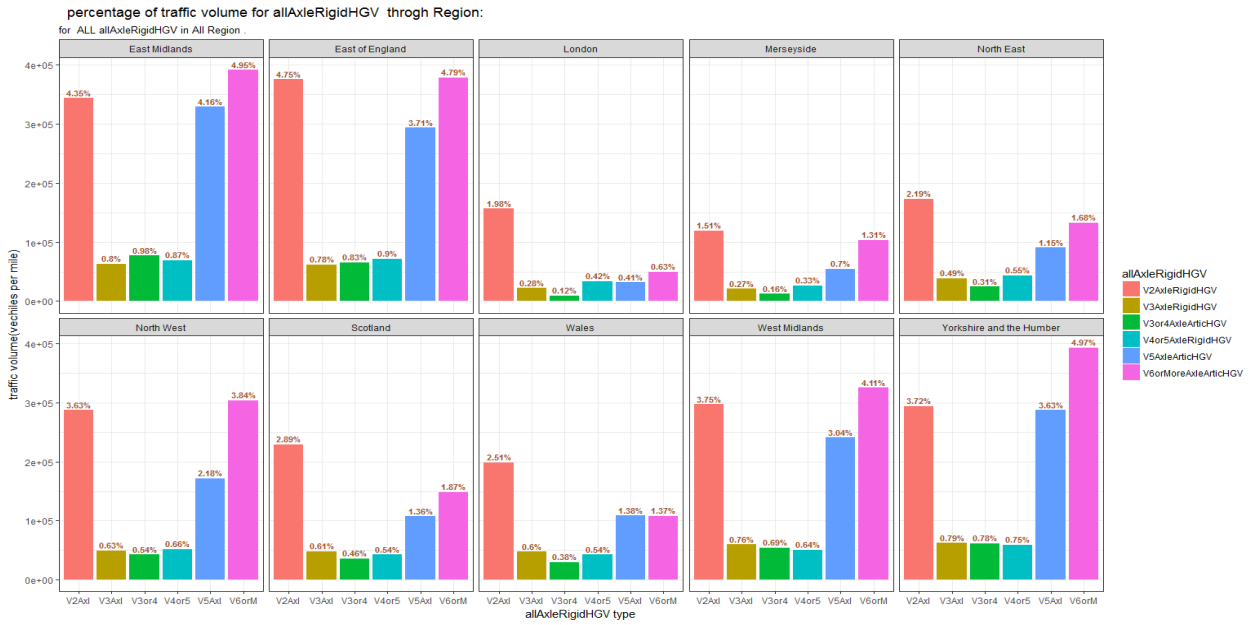


FIGURE7 79:PERCENTAGE OF TRAFFIC VOLUME FOR ALLAXLERIGIDHGV THROGH REGION

7.3 Shiny Web Implement

Shiny dashboard consists from three main parts:

Header
Sidebar

Body

7.3.1 Implementation of dashboard page:

```
    ## app.R ##

    library(shiny)

    library(shinydashboard)



    ui <- dashboardPage(

      dashboardHeader(),

      dashboardSidebar(),

      dashboardBody()

    )
server <- function(input, output) { }

shinyApp(ui, server) # to Run App.
```

7.3.2 Header

7.3.2 .1 Messages menu.

FIGURE7 80:MESSAGES MENU

Message menu.

Description: contain messages coming or received from cleaning team, analysis and visualization team or project manager.

Code Sample:

```
dropdownMenu(type = "messages",
 messageItem( from = "Analysis dept", message =
 "Sales are steady this month.",

   icon=),

  messageItem(from = "data analyst", message
  = "How do I visualization?", icon =
  icon("question"),time = "13:45"


    ),

 messageItem(from = "Support", message = "The new server is
ready.",

   icon = icon("life-ring"),

    time = "2014-12-01" )

      ),
```

```
dropdownMenu(type = "tasks", badgeStatus = "success",
taskItem(value = 95, color = "green", "Data cleaning"),
taskItem(value = 90, color = "aqua", "Data visualization"),
taskItem(value = 40, color = "yellow","shiny App"), taskItem(value
= 30, color = "red","Documenation"))
```

A message menu looks like this:

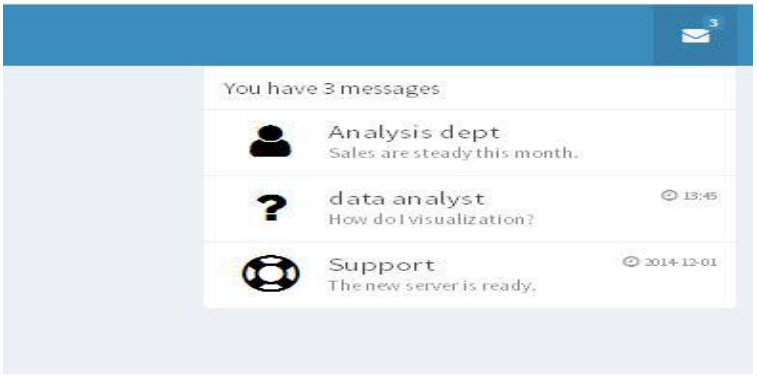


FIGURE7 81:MESSAGES

Task Menu

Description: Task menu that specify tasks progress.

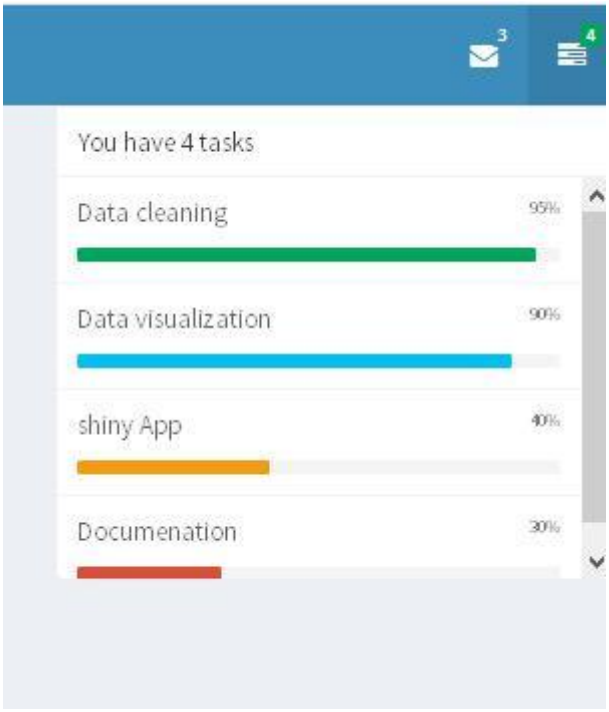
- Tasks are :
- Cleaning
- Transformation
- Exploration
- Visualization
- Modeling
- Documentation

Code implementation:

```
dropdownMenu(type = "tasks", badgeStatus = "success",
taskItem(value = 95, color = "green", "Data cleaning"),
taskItem(value = 90, color = "aqua", "Data visualization"),
taskItem(value = 40, color = "yellow","shiny App"),
taskItem(value = 30, color = "red","Documenation")

  )
```

Task menu Looks like this:



7.3.3 Sidebar

Description: A sidebar is typically used for quick navigation. It can contain menu items that behave like tabs in a tab Panel, as well as Shiny inputs, like sliders and text input.

Sidebar consist of

Sidebar Menu

Bookmarking and restoring selected tabs. Dynamic content.

Inputs in side bar.

7.3.3.1 Sidebar Menu

Description: Links in the sidebar can be used like tab Panels from Shiny. That is, when you click on a link, it will display different content in the body of the dashboard.

Consist of:

Menu items…Included in sidebar.

Menu items are put in sidebar Menu ()

Code sample:

```
sidebar <- dashboardSidebar(

  sidebarMenu(menuItem("Home", tabName = "home", icon = icon("home")),
```

```
menuItem("Datasets", tabName = "Datasets", icon = icon("table")),
```

```
menuItem("Data cleaning", tabName = "Data-cleaning", icon =
icon("th "), startExpanded = FALSE, menuSubItem("Documentation",
icon = icon(" file-text-o"), href = "index.html" , selected = T))
```

```
menuItem("Data Visualization", tabName = "Data-visual", icon =
icon(" th"), startExpanded = FALSE, menuSubItem("Documentation",
icon = icon ( "file-text-o"), href = "visual.html" , selected =
T)),
menuItem("charts", tabName = "gui" ,icon = icon("bar-chart-o"),
badge Label = "Moore..", badgeColor = "blue")))
```

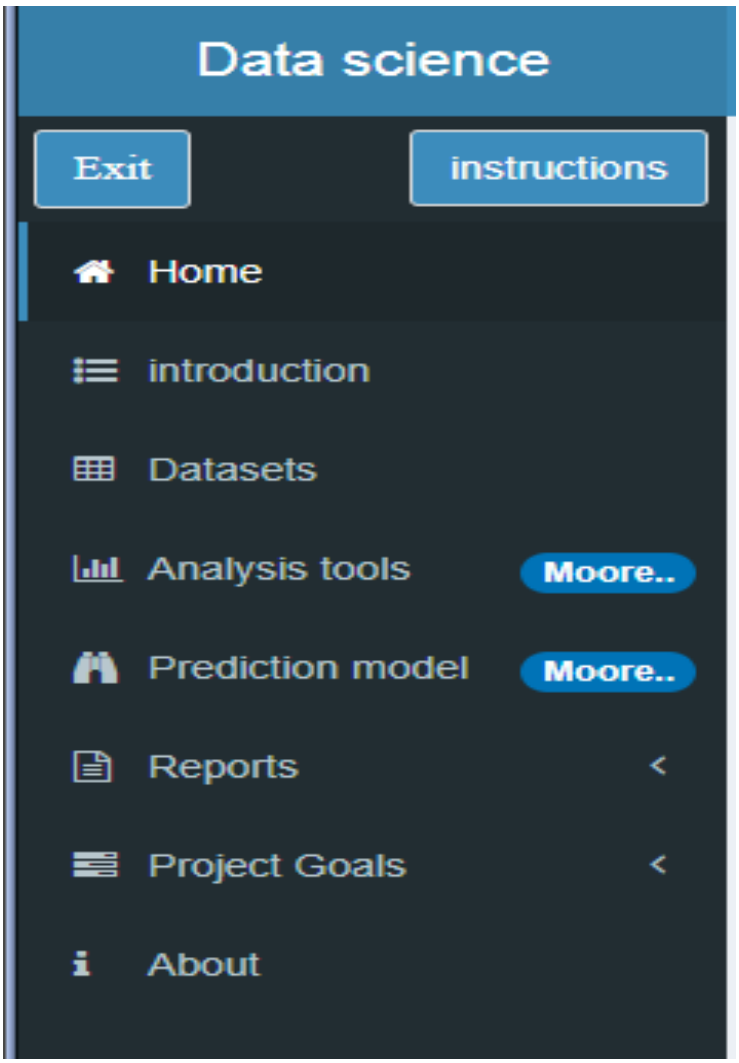


FIGURE7 83:MENU ITEMS

Tab items….included in body.

7.3.4 Body

Tab items….included in body.

7.3.4.1 Home page:

Description: the first page of app contain description about Data since filed

Code sample:

```
body<-dashboardBody(id="body",

tabItems(

tabItem( tabName="home",box(width=13, title = "Data science project", status = "primary",
solidHeader = TRUE, collapsible = TRUE, h1(class="h", imageOutput("image"),
p(class="p1","Data Analysis"),div(class="div1","Hellow in data Analysis pipeline")))
```

FIGURE7 84: HOME PAG

7.3.4.2 Uploading data sets page.

Description: contains two file uploading

1- First one for uploading traffic data.

2- Second one for uploading accidents data or any other data.

File upload controls are created by using functions inside tab item datasets.

Functions define UI for data upload page.

- fluidPage() → *define UI for data upload*
- titlePanel() → App title
- sidebarLayout() → with input output definitions.
- sidebarPanel() → for data inputs.
- fileInput() → select a file.
- checkboxInput ( ) → checkbox if file has a header.

- Main panel() → for displaying output.

Tabsetpanel(). Create tab set that contains

Tab Panel elements .Tab sets are useful for dividing output into multiple independently viewable sections.

Code sample: UI code.

tabItem(tabName = “Datasets”,

```
box(width=20,

title = "Uploding Data", status = "primary",solidHeader = TRUE,
 collapsible = TRUE,

  fluidPage(theme="style.css",

sidebarLayout(

 sidebarPanel(id="sidebar",

fileInput('file1', 'Choose traffic data:',
        accept=c('text/csv',

    'text/comma-separated-values,text/plain',
     '.csv')),uiOutput("fromCol1"),

fileInput('file2', 'Choose accident data:',
             accept=c('text/csv',

 'text/comma-separated-values,text/plain',
 '.csv')),uiOutput("fromCol2")

    mainPanel(width=3, # contain tab set panels

     tabsetPanel( # contain tabpanel.

       id = 'dataset',

        tabPanel(id='d',"traffic", DT::dataTableOutput("contents1
", width = 650),icon = icon("table")),

   tabPanel("accidents",DT::dataTableOutput("contents2", width =65
0),icon = icon("table"))
```

```
      ) ) ) )) ))

# end file uploading page.
```

Functions define Server for data upload app.

```
myData <- reactive({ inFile <- input 1 ( . ( )) ( ) < – .
( datapath, header = TRUE) data }) myData3<-myData myData2 <-reactive({ inFile <-
input 2 ( . ( )) ( ) 2 < – . ( datapath, header = TRUE)
data2})
```

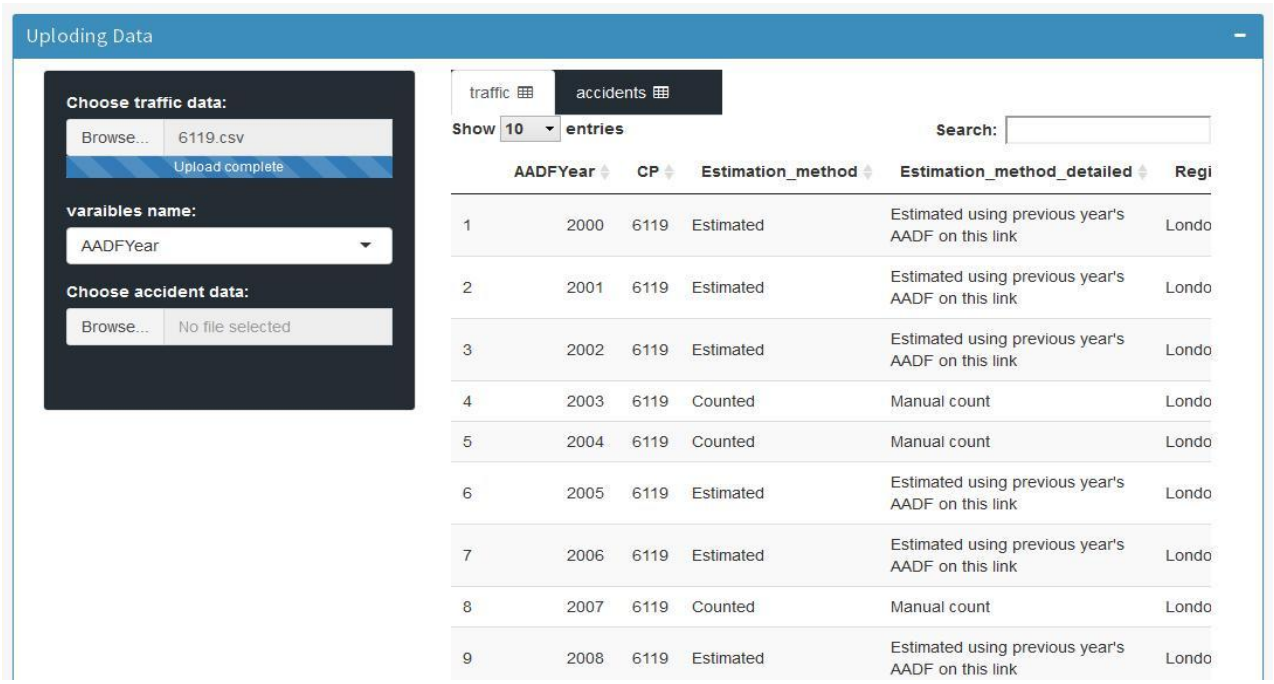


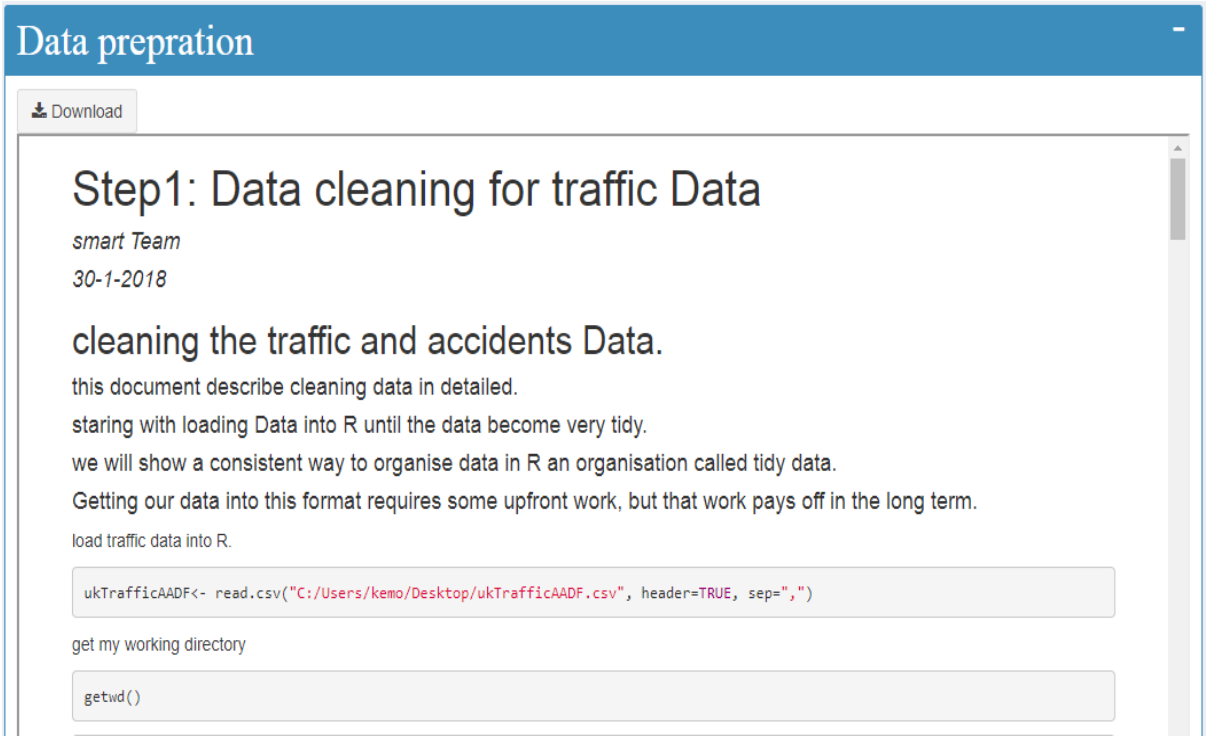
FIGURE7 85: UPLOADING DATA INTERFACE

7.3.4.3 Data cleaning page.

Description: Present user or cleaning team reports and Guidelines about how the accidents and traffic data were cleaned step by step.

Code sample

```
menuItem("Data cleaning", tabName = "Data-cleaning", icon = icon("th"), startExpanded
    = FALSE, menuSubItem("Documentation", icon = icon("file-text-o"), href = "index.html" ,
    selected = T))
```



7.3.4.4 Data visualization page.

Description: Present user or visualization team reports and Guidelines about how the accidents and traffic data were visualized and results.

Code sample:

```
menuItem("Data Visualization", tabName = "Data-visual", icon = icon("th"),
startExpanded = FALSE, menuSubItem("Documentation", icon = icon( "file-text-o"),
href = "visual.html" , selected = T

)
```

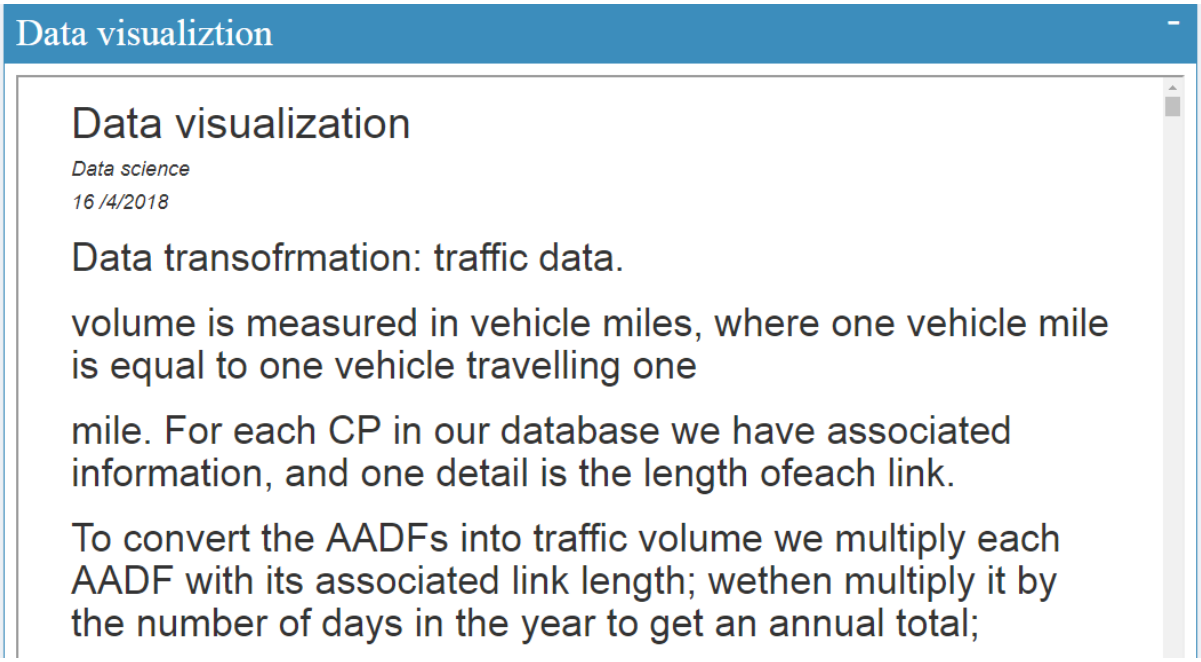


FIGURE7 87: DATA VISUALIZATION INTERFACE

7.3.4.5 Charts

Description: In charts element you can make graphics with analysis to datasets.

Charts tabitem contains four components:

Ggplot tab panel.

-ggplot package: A system for declaratively creating graphics, based on The Grammar of Graphics. You provide the data, tell ggplot2 how

to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

Plotly tab panel.

-plotly package: Easily translate 'ggplot2' graphs to an interactive web-based version and/or create custom web-based visualizations directly from R.

How to use ggplot and plotyly to make visualize to your data?

1-Select type of function:

Scatter plot:
Density plot:
Histogram:

Barplot:
Geombar:
Geompoint:

2-Names of variables you need to visualize theme and

3- Select if make grouping based on another variable ex.

Face row function.

Face column function
Grouping function.

The image show graph using ggplot and function

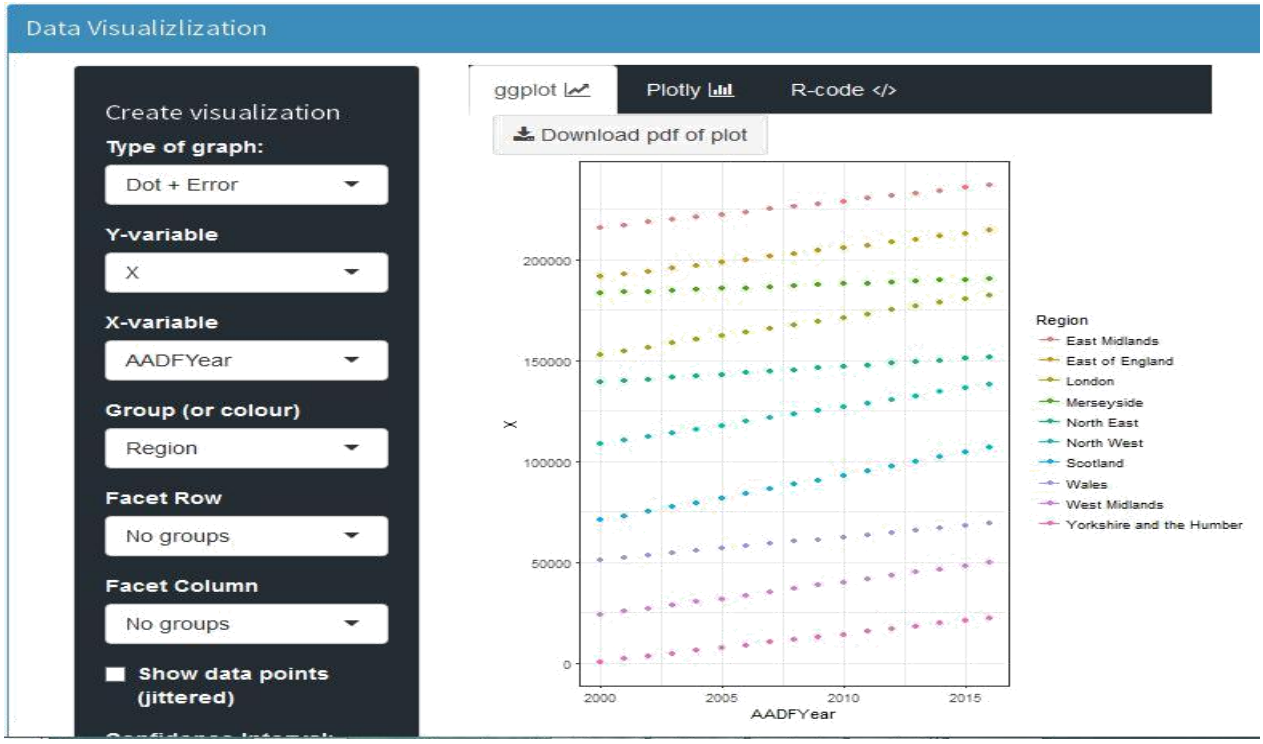


FIGURE7 88:GRAPH INTERFACE

Change aesthetics.

-You can change graphs:

- Color
- Font size
- Graph description

4- Size of graph when uploadin

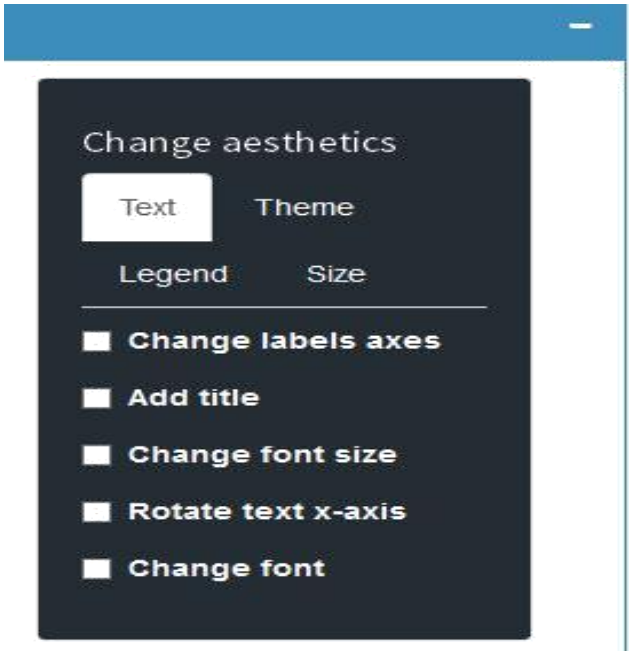


FIGURE7 90: CHANGE AESTHETICS

Graph code tab panel

- Provides the code of graph. Can copy it.

After the change and update the graph you can get the code of graph.

ggplot | Plotly | R-code </>

```
## You can use the below code to generate the graph.
## Don't forget to replace the 'df' with the name
## of your dataframe

# You need the following package(s):
library("ggplot2")

# The code below will generate the graph:
graph <- ggplot(df, aes(x = AADFYear, y = X, colour = R
egion)) +
  geom_point(stat = 'summary', fun.y = 'mean') +
  geom_errorbar(stat = 'summary', fun.data = 'mean_se',

          width=0, fun.args = list(mult = 1.96)) +
  labs(x = 'Year', y = 'Traffic volume.') +
  ggtitle('traffic flow.') +
  theme_bw()
graph

# If you want the plot to be interactive,
# you need the following package(s):
library("plotly")
ggplotly(graph)

# If you would like to save your graph, you can use:
ggsave('my_graph.pdf', graph, width = 14, height = 14,
units = 'cm')
```

FIGURE7 91:CODE OF GRAPH

8

chapter eight

Conclusion and future work

- conclusion
- Future work
- Refernces

8.1 Conclusion:

This project provide a lot of answer and solution for all problemes which asked to be solve.

By providing analysis for dataset and extract knowledge that help decision makers to avoid reasons which lead to problems at country and also provide interactive analysis tool to analysie yourdataset easly and predication for rate of accidents over future years

We provide solutions for all this problem

- Changing of traffic flow and it's impacted accidents
- predict accident rates over time
- Improve accident rates
- Identify infrastructure needs, failings and successes
- Rural and Urban areas differed
- Differences between England, Scotland, and Wales

The successful completion of graduation project is an indication of student's preparedness to pursue a professional career. At the same time the quality of graduation project in the collage reflect the academic achievement of computer science and information system faculaty at Beni Suef University.

8.2 Future work

The following goals we will study it next:

- Plot interactive maps of changing trends, e.g. how London has, changed for cyclists? Busiest roads in the nation?
- Which areas never change and why?
- North vs. South, East vs. West.
- And we will work on develop a complate system to manage traffic in city.
- Develop application that will help drivers in avoid the accident by take from him the condition that he will drive in it like(weather, light, type of way, night or day….,etc) based on this condition the data I have I will told him the rate of make accident or not.

Our Application Design:

1- Home page about application

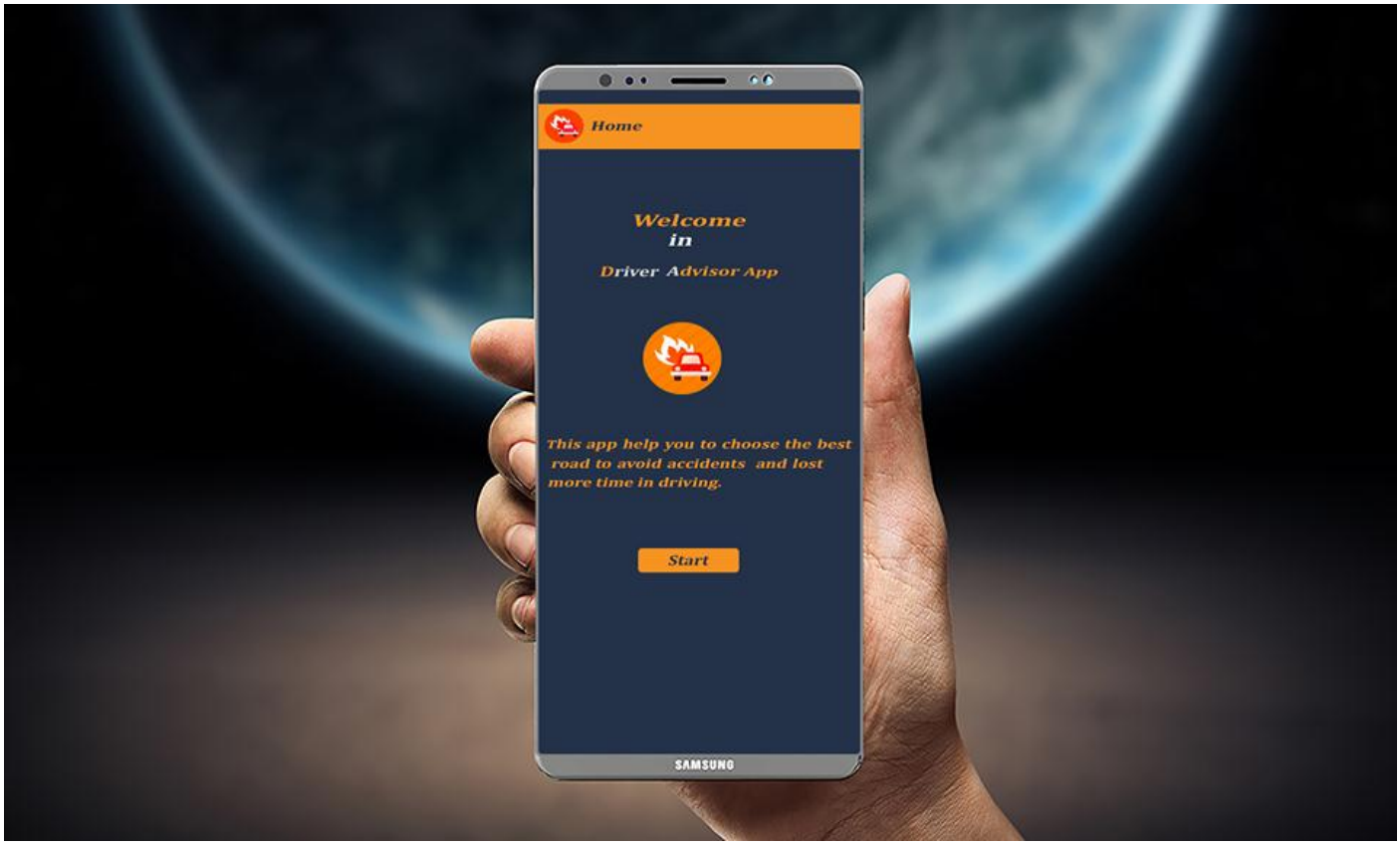


FIGURE8.1:HOME PAGE TO FUTURE APPLICATION

2- Login or sign up to our application.

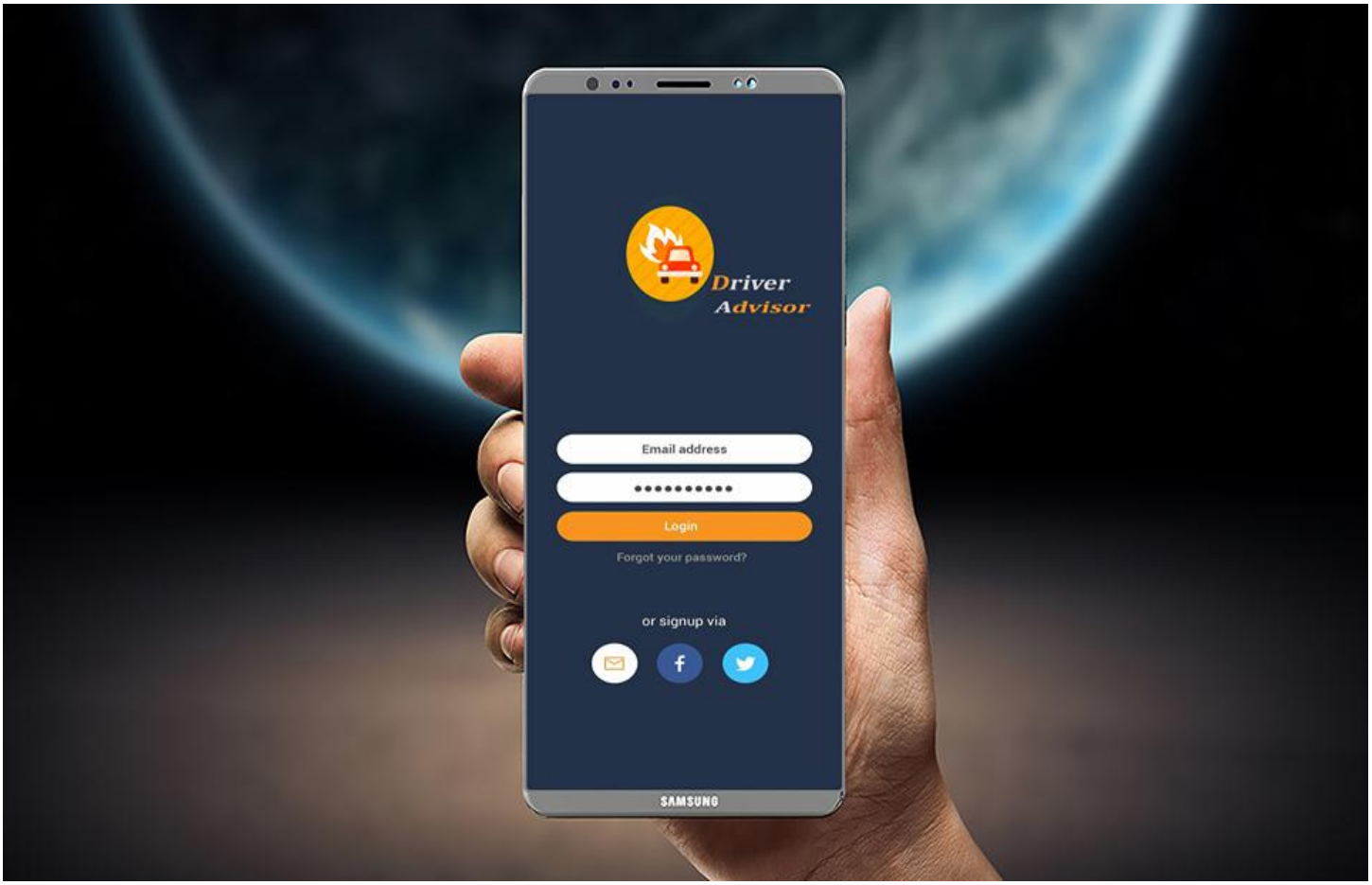


FIGURE 8 2: LOGIN OR SIGN UP TO OUR APPLICATION

3- Select the condition while you will drive like road type, speed, weather condition and vechiles type, based on this condition I will tell you my predication for you to make accident or not and tell you our advice.

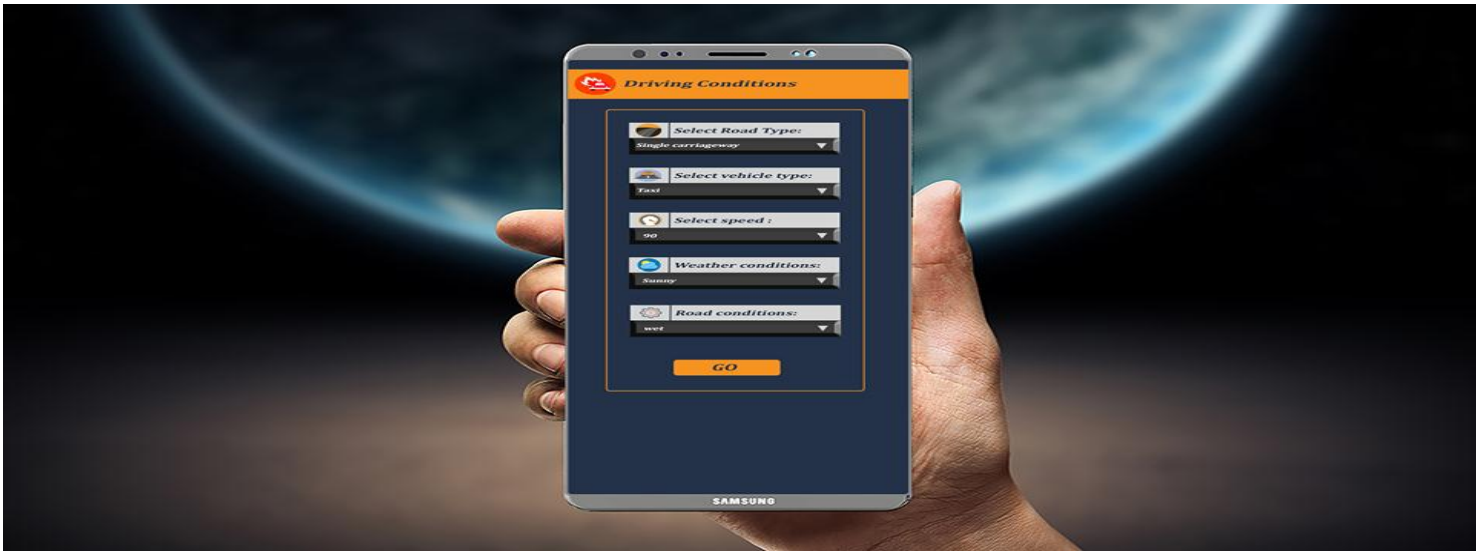


FIGURE 8 3: CONDITIONS FORM

4- Based on your selection it will move to next step that tell you our predication to make accident then click take our advice.

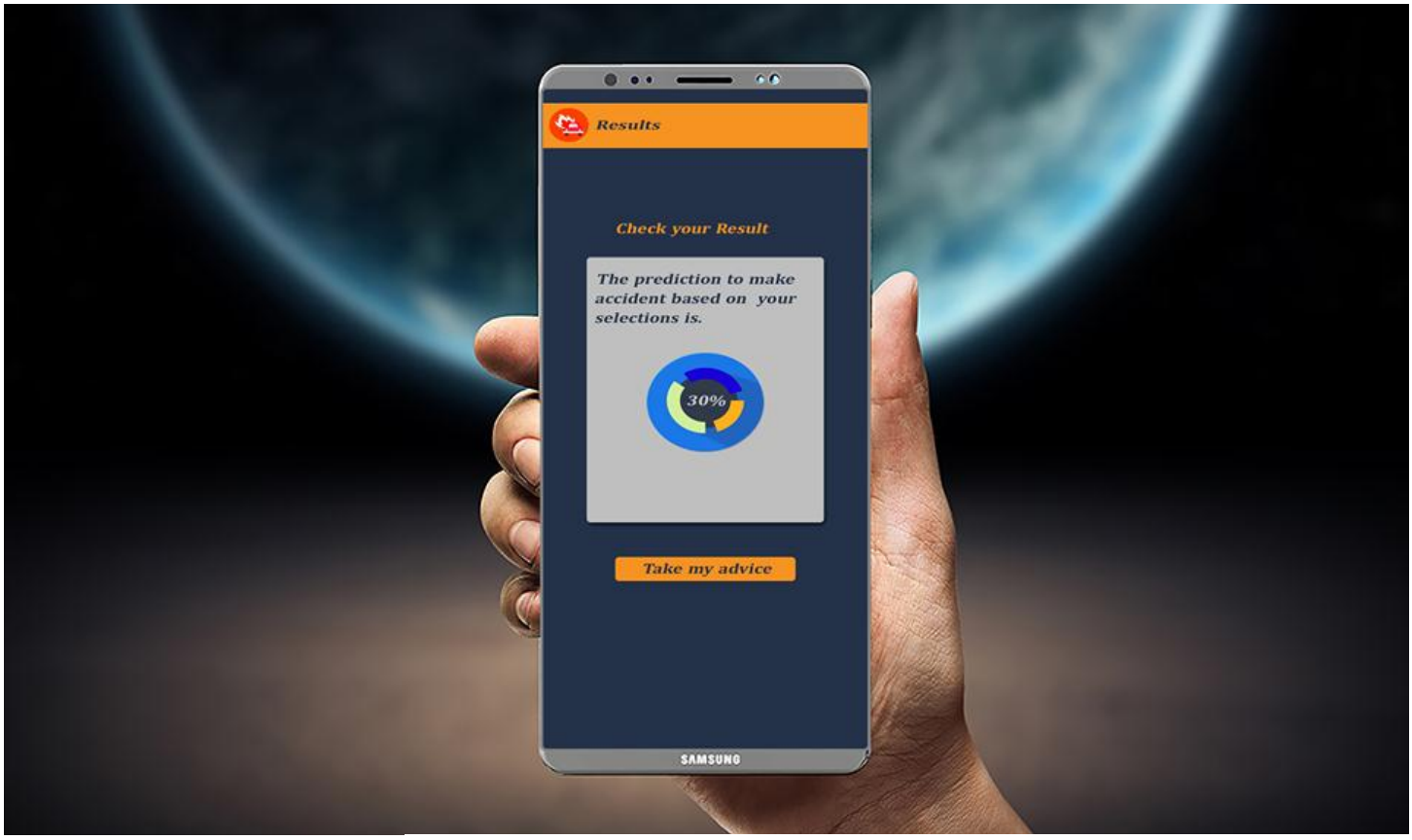


FIGURE 8 4: PREDICTION FORM

5- Here we will tell you the best conditions to avoid accident and to not loss your time.

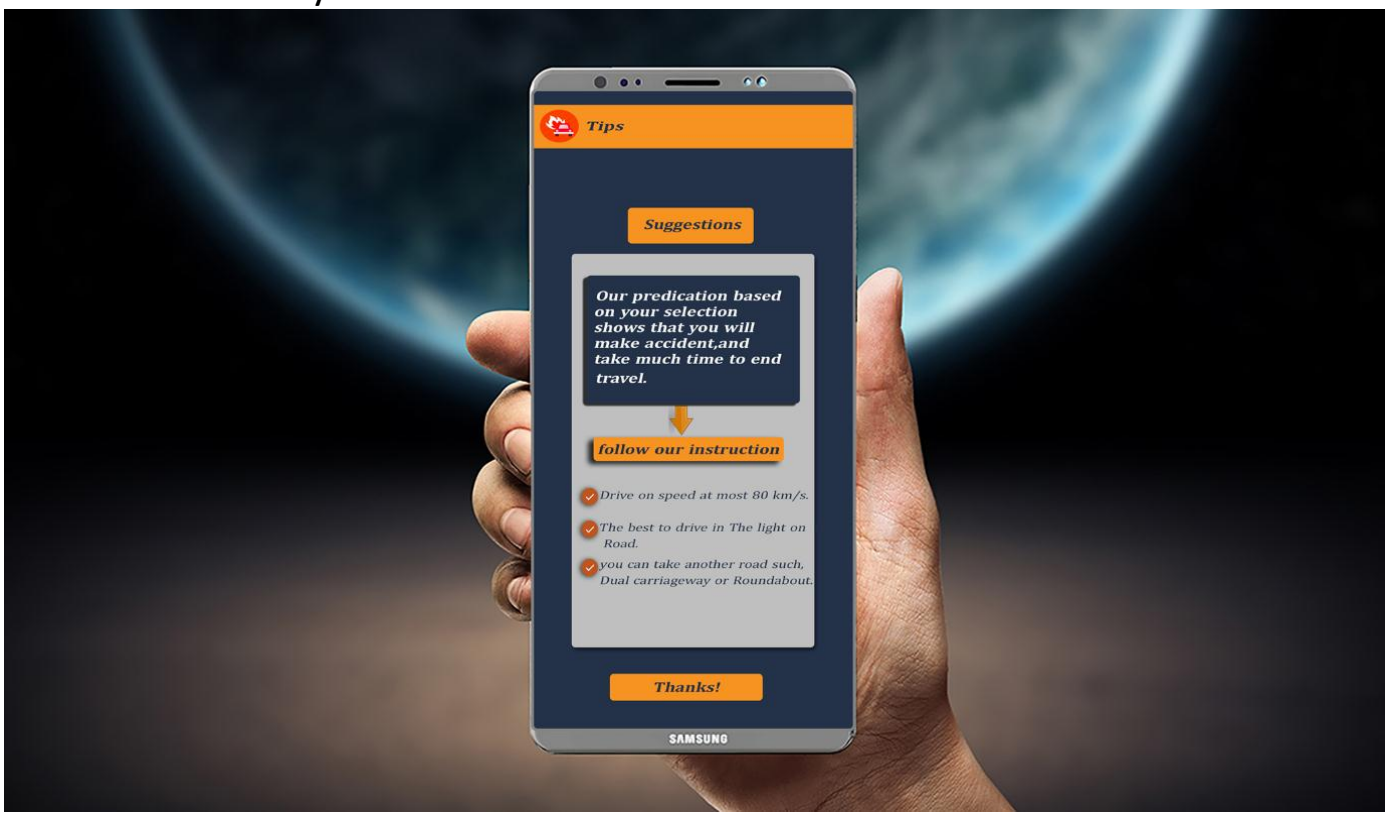


FIGURE 8 5:SUGGESTIONS

References:

[1] Amtrak. (n.d.). *Trend and Seasonality*. Retrieved from http://people.brandeis.edu: http://people.brandeis.edu/~blebaron/classes/fin250a/regression/trendSeason.html#amtrak-linear-trend

[2] connect, r. s. (2018, 5 23). *Combining Shiny & R Markdown*. Retrieved from R Studio Connect: https://beta.rstudioconnect.com/content/2671/Combining-Shiny-R-Markdown.html

[3] Connect, R. S. (2018, 5 24). *Combining Shiny & R Markdown*. Retrieved from R Studion Connect: https://beta.rstudioconnect.com/content/2671/Combining-Shiny-R-Markdown.html

[4] Connect, R. S. (n.d.). *shinydashboard*. Retrieved from R Studion Connect: https://rstudio.github.io/shinydashboard/appearance.html

[5] *Data Visualization Shiny Application.* (2017, 5 5). Retrieved from cl.indiana.edu: http://cl.indiana.edu/~obscrivn/Shiny-Introduction.pdf

[6] Gert Stulp [aut, c. (n.d.). *ggplotgui*. Retrieved from cran.r-project: https://cran.r-project.org/web/packages/ggplotgui/index.html

[7] ggvis. (n.d.). *Interactivity*. Retrieved from rstudio: https://ggvis.rstudio.com/interactivity.html

[8] Government, U. (2017, 9 25). *1.6 million UK traffic accidents*. Retrieved from Kaggle: https://www.kaggle.com/daveianhickey/2000-16-traffic-flow-england-scotland-wales/data

[9] Hyndman, R. J. (n.d.). *Regression with time series data*. Retrieved from otexts.org: https://www.otexts.org/fpp/4/8

[10] Majerus, R. (n.d.). *Bivariate Regression*. Retrieved from shiny: https://rich.shinyapps.io/regression/

[11] *RDocumentation*. (2017, 3 22). Retrieved from RDocumentation: https://www.rdocumentation.org/packages/Matrix/versions/1.2-14/topics/is.na-methods

[12] reliawiki. (2018, 5 25). *Simple Linear Regression Analysis*. Retrieved from reliawiki: http://reliawiki.org/index.php/Simple_Linear_Regression_Analysis

[13] *shiny*. (2017, 6 1). Retrieved from shiny.rstudio: http://shiny.rstudio.com/tutorial/

[14] shiny. (2018, 5 12). *Combining Shiny & R Markdown*. Retrieved from RStudio Connect: https://beta.rstudioconnect.com/content/2671/Combining-Shiny-R-Markdown.html

[15] shiny. (n.d.). *Interactive web applications with RShiny*. Retrieved from cyberhelp: http://cyberhelp.sesync.org/basic-Shiny-lesson/2016/07/26/

[16] *Simple Linear Regression Analysis*. (2018, 5 15). Retrieved from reliawiki.org: http://reliawiki.org/index.php/Simple_Linear_Regression_Analysis

[17] Teetor, P. (2011). *RCook Book.* Gravenstein Highway North: O'Reilly Media.

[18] Wickham, G. G. (2017). *R for data science.* Amazon website: O'Reilly January.